Entwurf und Modellierung einer universellen Telearbeitsumgebung auf Basis einer serviceorientierten Architektur

Dissertation

zur Erlangung des akademischen Grades Doktoringenieur (Dr.-Ing.)

vorgelegt an der
Technischen Universität Dresden
Fakultät Informatik

eingereicht von

Dipl.-Inform. Iris Braun
geboren am 27. Februar 1972 in Elsterwerda

am 11. August 2005

Gutachter:	Prof. Dr. rer. nat. habil. Dr. h. c. Alexander Schill	TU Dresden
	Prof. Dr. rer. oec. habil. Dr. h. c. Wolfgang Uhr	TU Dresden
	Prof. Dr. phil. Michael Jäckel	Universität Trier

Tag der Verteidigung: 1. November 2005

DANKSAGUNG

Die vorliegende Dissertation entstand während meiner Tätigkeit als Mitarbeiterin am Lehrstuhl Rechnernetze der Technischen Universität Dresden sowie als Stipendiatin eines Wiedereinstiegsstipendiums des Freistaates Sachsen. Viele Personen haben auf unterschiedliche Weise zum Zustandekommen und Gelingen dieser Arbeit beigetragen, denen ich an dieser Stelle herzlich danken möchte.

Besonderer Dank gilt meinem Doktorvater Prof. Alexander Schill für die fachliche und persönliche Unterstützung, das entgegengebrachte Vertrauen, die motivierenden Worte in schwierigen Phasen und die unbürokratische Organisation meiner Finanzierung. Darüber hinaus möchte ich ihm dafür danken, dass er es mir ermöglichte, meine Arbeit auf zahlreichen internationalen Konferenzen und Tagungen einem breiten Fachpublikum zu präsentieren.

Weiterhin möchte ich Prof. Wolfgang Uhr und Prof. Michael Jäckel danken, dass Sie die vorliegende Arbeit begleitet und durch Ihre Anregungen befruchtet haben. Ebenso danke ich meinen Kollegen am Lehrstuhl Rechnernetze insbesondere Dr. Sabine Kühn und Dr. Thomas Springer für die gewinnbringende fachliche Diskussion meiner Ideen und die gute Zusammenarbeit.

Ganz herzlich danke ich auch meinem Partner Thomas Lauke, meinen Eltern und Schwiegereltern für die moralische Unterstützung und, dass sie mir den Rücken frei gehalten und sich so liebevoll um meinen Sohn Karl gekümmert haben, der es immer wieder geschafft hat, mich auch in schwierigen Momenten zu erheitern.

Auch meiner Freundin Franziska Füssel gilt mein Dank, da sie als außerfachliche Lektorin zur Allgemeinverständlichkeit der Arbeit beigetragen hat. Vergessen möchte ich auch nicht Josef Spillner, der als studentische Hilfskraft wesentlich an der Implementierung mitgewirkt hat, und alle Studenten, die im Projektumfeld ihre Beleg-, Bachelor- oder Diplomarbeit angefertigt haben.

INHALTSVERZEICHNIS

KAPITEL 1
EINLEITUNG

"Das größte Risiko der Informationsgesellschaft ist,
deren Chancen nicht zu nutzen."

Bangemann-Bericht, 1994 [EuR94]

1.1 Ausgangssituation und Motivation

Das Zusammenwachsen von Computern, Medien und Kommunikationsnetzen beeinflusst das Wirtschafts- und Privatleben heute wie nie zuvor. Die unaufhaltsamen Innovationen prägen uns ihren Stempel und ihren Rhythmus auf - am Arbeitsplatz wie auch zu Hause [BFH+99]. Um mit diesen rasanten Entwicklungen Schritt halten zu können, ist die Bereitschaft zu flexiblen Arbeitsmodellen unabdingbar geworden. Dazu bieten die weltweiten Netze inzwischen gute Bedingungen. Vielfältige Arbeitsaufgaben lassen sich mit modernen Telediensten bewerkstelligen - virtuelle Arbeitsumgebungen entstehen. Der Wandlungsprozess, der mit der zunehmenden Integration von Kommunikations- und Informationstechnologie einhergeht, wirkt sich sowohl auf die Arbeitswelt und Unternehmenskultur als auch auf unsere Gesellschaft aus. Unternehmen, die sich dieser „informationellen Revolution“ verschließen, werden auf den Märkten der Zukunft keine Chance mehr haben [ReG98]. Der Übergang in die moderne Informationsgesellschaft mit einem wissensorientierten Wirtschaftswachstum erfordert die Schaffung von neuen Arbeits- und Organisationsformen. Telearbeit ist dabei der Schlüssel zu einer flexibleren Arbeitsplatz- und Arbeitszeitgestaltung und bietet vielfältige Möglichkeiten und Vorteile [HeB97].

Ca. 17 % der deutschen Arbeitnehmer gehören zu einer der weltweit am stärksten wachsenden Berufsgruppen: den Telearbeitern. Nach zahlreichen Kontroversen über die Vor- und Nachteile haben vor allem die Entwicklung kostengünstiger Informations- und Kommunikationstechniken, aber auch der Handlungsdruck einer zunehmend globalisierten Wirtschaft das Thema Telearbeit in den letzten Jahren neu belebt. In den Jahren 1999-2002 hat sich die Anzahl der Telearbeiter in Deutschland fast verdreifacht (siehe Abbildung 1-1 [Kor02]). Auch die Politik hat die strategische Bedeutung von Telearbeit und der einhergehenden Wandlung zur Informationsgesellschaft längst erkannt. Bereits 1994 erarbeitete der Europäische Rat ein Rahmenpapier mit konkreten Vorschlägen für den Weg in die Informationsgesellschaft, den so genannten „Bangemann-Bericht“ [EuR94].

Dank der rasanten Entwicklung der elektronischen Medien und insbesondere der weltweiten Vernetzung von Computersystemen kann Telearbeit schon in naher Zukunft eine alltägliche Form des Arbeitens für viele geworden sein - zu Hause oder unterwegs, in einem "Telecenter" oder einem "virtuellen Büro" [BHS98a]. Um den Markt für Telearbeit zu beurteilen, kann man von einem geschätzten Gesamtvolumen von 13 Mio. möglicher Telearbeitsplätze in Deutschland ausgehen [Kor02]. Das Potenzial für Telearbeit ist somit nach wie vor erheblich größer als die derzeitige Praxis. Da eine präzise allgemeingültige Definition für Telearbeit in der Literatur aber fehlt, kann die Anzahl der tatsächlichen und potenziellen Telearbeitsplätze in den verschiedenen empirischen Untersuchungen erheblich variieren. Die Ergebnisse solcher Untersuchungen müssen daher immer im Zusammenhang mit der jeweiligen Definition interpretiert werden.

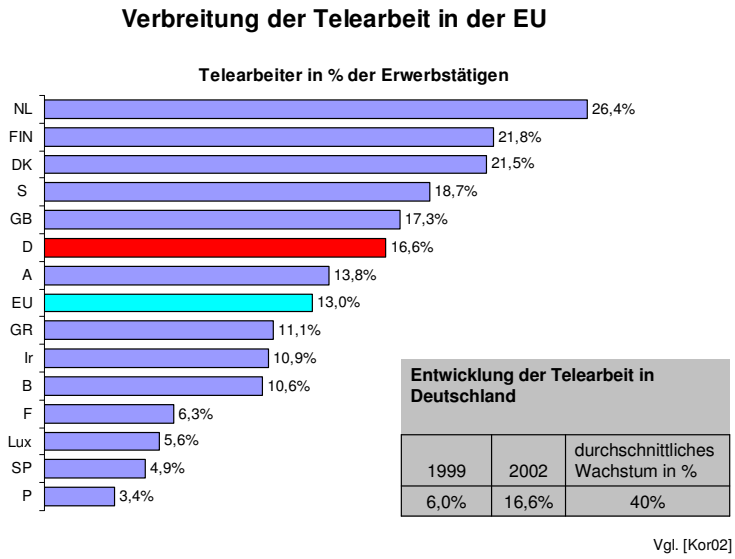

Abbildung 1-1: Verbreitung der Telearbeit in Europa und Deutschland

Herwig Heckl beschreibt die Entwicklung von Telearbeit in seinem Beitrag „Telearbeit aus Sicht der IT-Industrie“ [Hec95] wie folgt: „Produktivitätsdruck einerseits und die kostengünstige Verfügbarkeit der Technik andererseits führen im Augenblick in allen Industrienationen flächendeckend und spartenübergreifend zu Business-Reengineering-Maßnahmen von bisher nicht gekannten Ausmaßen, die alle konventionellen, papierorientierten Arbeitsprozesse in den Wirtschafts- und Verwaltungsbereichen in Frage stellen. In diesem Umfeld wird die Telearbeit nach langen Jahren schwieriger Grundsatzdiskussionen eine Schlüsselfunktion in allen neu entstehenden vernetzten Arbeitsprozessen übernehmen. Elektronisch gestützte Telearbeit mit ihren vielen denkbaren Varianten, von der Teleheimarbeit über Telezentren bis hin zu mobilen Büros, birgt für alle Beteiligten ein erhebliches ökonomisches Potenzial, mögliche negative Nebenwirkungen können von vornherein durch einen pragmatischen Abstimmungsprozess zwischen allen Beteiligten ausgeschaltet werden.“

Telearbeit wird dabei vor allem durch die räumliche Trennung des Arbeitsplatzes vom Standort des Unternehmens und die Nutzung von elektronischen Medien zur Überbrückung dieser Trennung charakterisiert. Eine detaillierte Definition des Begriffs „Telearbeit“ wird in Kapitel 2.1 der vorliegenden Dissertation erarbeitet. Das entfernte und kooperative Bearbeiten einer Aufgabe oder Dienstleistung durch einzelne oder mehrere Personen birgt eine Vielzahl technischer und organisatorischer, aber auch sozialer und arbeitswissenschaftlicher Herausforderungen in sich. Alle am Arbeitsprozess beteiligten Komponenten wie Arbeitsgegenstand, Arbeitsmittel, Hilfsmittel, Aufgaben, Dokumente, Informationen, Datenbestände, Bearbeiter, Auftraggeber, Auftragnehmer, Kunde usw. stehen in vielfältigen Beziehungen zueinander. Durch die mögliche weiträumige Verteilung dieser Komponenten wird es notwendig, sowohl räumliche als auch zeitliche Trennungen der am Arbeitsprozess beteiligten Elemente zu überwinden, um deren Zusammenwirken zu ermöglichen. Geschieht dies mit Hilfe von Informations- und Telekommunikationstechniken, kann von Telearbeit gesprochen werden. Hinter dem Begriff "Telearbeit" verbirgt sich eine Vielzahl von verschiedenen Arten der Arbeitsorganisation in Abhängigkeit von Arbeitsinhalt, zeitlichem und rechtlichem Rahmen der Beschäftigung, technischen Möglichkeiten der Telekommunikation und der Qualifikation des Arbeitnehmers.

Damit diese Arbeitsform ihr Produktivitätspotenzial ganz entfalten kann, gilt es jedoch, Telearbeitern ein flexibles Arbeitsumfeld zur Verfügung zu stellen und Berührungsängste mit den neuen Techniken zu überwinden. Heutige Informations- und Kommunikationstechnologien haben einen Stand erreicht, der im Büroumfeld dezentralisiertes Arbeiten und gleichzeitig enge Kommunikationsbeziehungen über große Entfernungen ermöglicht. Die notwendige Technik ist so weit entwickelt, dass Informationsverarbeitung und Kommunikation nahezu überall möglich ist [Hec95].

Im Rahmen vorangegangener Projekte wurden vielfältige Studien zum Thema Telearbeit durchgeführt. Bei der bisherigen Umsetzung von Telearbeit in der Praxis ergaben sich vor allem Probleme im Umgang mit den verschiedenen Arbeitsumgebungen im Büro und am Telearbeitsplatz sowie dem Abgleich der Arbeitsstände und -ergebnisse. Dabei konnten wir feststellen, dass die verwendeten Kommunikationstechnologien, der Fernzugriff auf Dokumente und Anwendungen, ein leichter Zugang zu Kollaborationsumgebungen und gemeinsame Arbeitstechnologien ein kritisches Element im Bezug auf den Erfolg der Telearbeit sind. Um eine hohe Transparenz der Arbeitsabläufe zu gewährleisten, müssen die realen Abläufe im Büro (z.B. Sekretariat, Technischer Support, Dienstbesprechungen) auf Dienste des Internets abgebildet und flexibel eingesetzt werden.

1.2 Ziele der Arbeit

Ziel der Arbeit ist die Modellierung und prototypische Entwicklung einer webbasierten, flexiblen und modular aufgebauten Arbeitsumgebung, die eine effektive Anpassung an die Arbeitsweise des jeweiligen Telearbeiters und Unternehmens ermöglicht. Dabei soll die Flexibilität der Lösung in der Weise erhöht werden, dass Telearbeit möglichst orts-, zeit- und geräteunabhängig erfolgen kann. „*Office is where you are*“ – diese Anforderung muss in erster Linie durch innovative informationstechnische Konzepte umgesetzt werden. Dabei geht es vorrangig um die Anpassung der software-technischen Unterstützung von Telearbeit an menschliches Arbeitshandeln und nicht umgekehrt.

Als hauptsächliche Designziele bei der Entwicklung der eigenen Lösung wurden Folgende festgelegt:

- flexible Konfiguration und Anpassung der Arbeitsumgebung,
- einfache Bedienbarkeit (wichtig für informatikfremde Nutzergruppen),
- Erreichbarkeit von überall unter Einsatz verschiedenster Netztechnologien,
- Transparenz der Umgebung und der verwendeten Dienste,
- Integration verschiedenster Anwendungen in einer Oberfläche.

Zu den wichtigsten Ansprüchen der Telearbeiter gehören vor allem eine schnelle Kommunikation mit Kollegen und Kunden und eine gute technische Betreuung. Aus technischer Sicht muss eine geeignete Unterstützung der entfernten Arbeit und der Gruppenarbeit realisiert werden, die die Aspekte Kommunikation, Kollaboration und Koordination abdeckt.

Im Gegensatz zu vorhandenen, meist über verschiedene Middleware-Lösungen durchgeführten punktuellen Integrationslösungen wird nach einer umfassenden universellen Lösung gesucht, welche bestehende Eigenentwicklungen, standardisierte Softwarekomponenten sowie Neuentwicklungen gemeinsam integriert. Ziel ist die automatische Auswahl, Verbindung und Interoperation von geeigneten Diensten, um alle Aufgaben des Telearbeiters zu lösen. Aufgrund der Vielzahl der in den Unternehmen anzutreffenden heterogenen Einzelsysteme stellt die Integration aller zur Unterstützung der Telearbeit relevanten Systeme eine große Herausforderung dar. Mit der Anzahl der einzubindenden Dienste und Backend-Systeme steigt auch die Komplexität der Integrationslösung.

1.3 Lösungsansatz

Mit der Entwicklung der Web-Service-Technologie wurde eine universelle Schnittstelle zur Kopplung von verschiedensten Anwendungen über das Internet geschaffen. Durch Web-Service-Infrastrukturen lassen sich Dienste aller Art von jedem Ort und nicht nur auf stationären Computern nutzen. Durch die "Schlankheit" der zugrunde liegenden Technologien unterstützen sie nahezu beliebige Endgeräte, einen Internet-Zugang vorausgesetzt. Die Technologie der Web Services verspricht einen mit Hilfe von XML (*eXtensible Markup Language*) definierten Standard-Kommunikationsmechanismus zwischen verteilten, lose gekoppelten heterogenen Anwendungen und ist damit gut für den Telearbeitskontext geeignet.

Web Services bieten die Möglichkeit, heterogene Dienste und Anwendungen plattformunabhängig in einer webbasierten Umgebung zu integrieren. Ebenso lassen sich bisher isolierte Dienste zu ganzen netzweiten Workflows komponieren. Dies eröffnet neue Möglichkeiten nicht nur für Telearbeit sondern auch für B2B- und B2C-Anwendungen[1]. Web-Service-Standards wie SOAP[2] oder WSDL (*Web Service Description Language*) haben sich innerhalb kürzester Zeit etabliert. Sie werden von den wichtigsten Softwareunternehmen unterstützt und im Rahmen unabhängiger Standardisierungsgremien wie W3C[3] und OASIS[4] weiterentwickelt. Neben diesen etablierten Standards existieren mittlerweile viele weiterführende Standardisierungsinitiativen für die Bereiche Sicherheit, Komposition, Management und Interoperabilität, die für die Integration von Web Services von Bedeutung sind [QuW03].

Aufbauend auf den definierten Anforderungen werden die benötigten Dienste im Telearbeitsszenario festgelegt und klassifiziert und eine Architektur der Telearbeitsanwendung entworfen. Dabei werden die Paradigmen der SOA (*Service Oriented Architecture*) - also die Verwendung verteilter, lose gekoppelter Dienste - zugrunde gelegt. Durch die Bildung von wiederverwendbaren Diensten aus Anwendungen und Informationsobjekten können die Komponenten der Telearbeitsumgebung modular zusammengestellt und daher auch schnell verändert werden. Somit können die bereitgestellten Inhalte, Dienste und Funktionen beliebig integriert und zudem nutzerspezifisch angepasst werden. Das über einen Webbrowser zugängliche Telearbeitsportal führt die einzelnen Dienste für die Anwender zu einem einheitlichen System zusammen und bietet ihnen einen rollenbasierten und transparenten Zugriff auf alle Informationen und Anwendungen, die sie zur Erledigung ihrer Aufgaben benötigen, ohne die Quelle der Informationen oder die zugrunde liegenden Backend-Systeme kennen zu müssen.

Für die weitere Arbeit wurden folgende Thesen formuliert:

1. Mit der Umsetzung einer serviceorientierten Architektur auf Basis von Web Services lässt sich eine Telearbeitsumgebung schaffen, die im Vergleich zu vorhandenen Ansätzen flexibler konfigurierbar und damit universeller einsetzbar ist.

---

[1] B2B (Business to Business) steht für (elektronische) Kommunikationsbeziehungen zwischen Unternehmen, B2C (Business to Customer) dagegen für die Beziehungen zwischen Unternehmen und Konsumenten.

[2] Ab Version 1.2 (Juni 2004) wird „SOAP“ nur noch als Eigenname benutzt. Die bisherige Verwendung als Akronym für „Simple Object Access Protocol“ entfällt damit.

[3] Das WWW-Konsortium (auch W3C genannt) ist eine non-profit Standardisierungsgesellschaft, deren Hauptziel die Weiterentwicklung des Internet-Standards und verwandter Protokolle ist.

[4] Die Organization for the Advancement of Structured Information Standards (OASIS) ist eine internationale, nicht-gewinnorientierte Organisation, die sich mit der Weiterentwicklung von Web-Service-Standards beschäftigt.

2. Komplexe Geschäftsprozesse und Arbeitsabläufe können durch die Komposition von Telearbeitsdiensten aus wiederverwendbaren Basisdiensten abgebildet werden. Dies führt zu kürzeren Entwicklungszeiten und Kostenvorteilen.
3. Mit Hilfe der generischen Beschreibung der Benutzerschnittstellen von Web Services kann eine flexible Integration der Telearbeitsdienste in ein Portal erfolgen und dem Telearbeiter damit eine einheitliche Benutzeroberfläche für alle benötigten Dienste und Anwendungen bereitgestellt werden.

1.4 Gliederung der Arbeit

Zu Beginn werden in Kapitel 2 verschiedene Definitionen von Telearbeit analysiert und die besonderen Spezifika dieser Arbeitsform erörtert. Ausgehend von der allgemeinen Betrachtung der verschiedenen Formen und Rahmenbedingungen werden konkrete Anwendungsszenarien für Telearbeit beschrieben. Darauf aufbauend werden die Anforderungen an eine universelle Arbeitsumgebung festgelegt.

In Kapitel 3 werden vorhandene Softwarelösungen auf Ihre Nutzbarkeit im Rahmen von Telearbeit untersucht und klassifiziert. Es existiert eine Fülle von kommerziellen Anwendungen, die unter bestimmten Voraussetzungen für Telearbeit genutzt werden können, z.B. Groupware- oder Workflow-Applikationen. Dabei werden die in Kapitel 2 festgelegten Anforderungen auf diese Lösungen abgebildet, um eine Abgrenzung der eigenen Arbeit gegenüber anderen Ansätzen und Systemen vornehmen zu können.

In Kapitel 4 werden die Basisprotokolle und grundlegenden Technologien zur praktischen Umsetzung serviceorientierter Architekturen spezifiziert und auf Ihre Nutzbarkeit im Telearbeitskontext untersucht.

Ausgehend von der Analyse der für Telearbeit relevanten Anforderungen an eine universelle Software-Lösung wird in Kapitel 5 eine Architektur der Telearbeitsanwendung entworfen. Es werden die notwendigen Basisdienste festgelegt, die eine Telearbeitsumgebung zur Verfügung stellen sollte, und entsprechend ihrer Aufgaben klassifiziert. Danach erfolgt die Abbildung der Telearbeitsdienste auf die Web-Service-Technologie. Dabei werden noch offene Fragen zur Umsetzung der einzelnen Dienste als Web Services geklärt und Empfehlungen für die Implementierung gegeben. Weiterhin wird die flexible Integration der Dienste in die Telearbeitsumgebung auf zwei Ebenen beschrieben. Die prozessorientierte Integration dient der Abbildung von Arbeitsabläufen und Geschäftsprozessen auf komplexe Dienste. Um dem Telearbeiter eine einheitliche Sicht auf alle genutzten Dienste zu ermöglichen, erfolgt anschließend eine benutzerorientierte Integration der Dienste in ein Portal.

Im Rahmen der abschließenden Evaluierung in Kapitel 6 sollen die Vor- und Nachteile der eigenen Lösung überprüft werden. Mit einer prototypischen Implementierung und Emulation einzelner Telearbeitsdienste und der Integration in das Telearbeitsportal wird die Funktionsweise der vorgeschlagenen Lösung nachgewiesen. Durch eine Kapselung der verschiedenen Anwendungen kann dies für einzelne Dienste durchgeführt werden. Weiterhin wird der Einsatz der Lösung in verschiedenen Telearbeitsszenarien untersucht, um somit die Praxistauglichkeit der Lösung zu bewerten. Zum Abschluss der Validierung wird die entwickelte Lösung dahingehend bewertet, wie sie den anhand der in Kapitel 2 definierten Anforderungen aufgestellten Kriterienkatalog erfüllt.

Abschließend werden in Kapitel 7 die Ergebnisse der Arbeit zusammengefasst und weitere Entwicklungs- und Ausbaumöglichkeiten des Konzeptes vorgestellt.

KAPITEL 2
PROBLEMSTELLUNG UND ANFORDERUNGSANALYSE

„Die industrielle Revolution verlagerte die Arbeit von Feld und Heim in die Fabrik. Das Internet bringt sie dorthin zurück, wo wir angefangen haben."

Michael J. Cunningham, 2001 [Cun01]

2.1 Definition von Telearbeit

Mitte der 70er Jahre bekam die Diskussion über die Auslagerung von Arbeit durch die Nutzung moderner Telekommunikationstechnologien erstmals großen Auftrieb. Unmittelbarer Auslöser dafür war die Ölkrise 1973. Man suchte Auswege aus der Mobilitätskrise, die durch die spürbare Verteuerung von Treibstoff hervorgerufen worden war. Jack Nilles prägte im Jahr 1973 erstmals den Begriff „Telecommuting" (Telependeln) [Nil76], der durch Alvin Toffler in „*The Third Wave*" [Tof80] populär gemacht wurde. Der Fokus lag für Nilles auf den entstehenden Einsparungspotenzialen beim Pendelverkehr sowie den räumlichen Aspekten der Telearbeit. Bei seinem Konzept standen zunächst weniger die Potenziale von dezentralisierter, computerisierter Arbeit im Vordergrund als vielmehr eine „verkehrs-, energie-, städtebau- und raumordnungspolitische Neuorientierung" [Kir96]. Später erhoffte man sich von der Einführung moderner Kommunikationstechnologien Möglichkeiten, Arbeitsplätze zu dezentralisieren, Städte zu vitalisieren und die Arbeitsbedingungen in den Firmen zu verbessern.

Der Grundgedanke des „*Telecommuting*" war dabei, dass nicht die Beschäftigten, sondern nur die arbeitsrelevanten Daten zwischen Wohnung und Firmenzentrale pendeln sollten [Nil94]. Dieses Konzept wurde später um den Gedanken des „*remote office work*" erweitert. Wenn die entsprechende Telekommunikations-Infrastruktur vorhanden ist, kann praktisch jeder Ort als Arbeitsplatz dienen. Ob Wohnung, Zug, Auto oder Büroräume in der Nähe der Wohnung, überall kann man sein Arbeitspensum erfüllen [Kir96]. Auf dem Weg zu einer globalisierten Wirtschaft wurde so auch die Arbeit immer mehr unabhängig von Ort und Zeit, das Prinzip von „*Anytime-Anyplace*" [RMS+00] wird in immer größerem Maß auch für die Arbeit gültig. Gerald Rauscher [Rau00], Arbeitszeit-Referent der Hewlett-Packard Deutschland GmbH, beschreibt die Entwicklung wie folgt: „Durch eine strukturelle Verlagerung von Arbeit in Projekte, Prozesse und Problemlösungen sowie durch den Einsatz moderner Kommunikationsmedien verlieren vordefinierte Anwesenheitszeiten und feste Einsatzorte immer mehr an Bedeutung."

Telearbeit ist dabei kein spezielles Berufsbild, sondern eine Organisationsform der Arbeit. Bei der Eingrenzung des Begriffes Telearbeit stellt sich daher nicht die Frage nach den Arbeitsaufgaben selbst, sondern auf welche Art und Weise die durchaus sehr unterschiedlichen Aufgaben erledigt werden. Dies signalisiert schon die Bezeichnungsvielfalt, die aufgrund der verschiedensten Arbeitsweisen oft synonym für Telearbeit verwendet werden: Teleworking, Telecommuting, Telependeln, elektronische Fernarbeit, Bildschirm-Heimarbeit, Teleheimarbeit, außerbetriebliche Arbeitstätigkeit, Computerheimarbeit und Homejobbing.

Für Telearbeit gibt es sehr unterschiedliche Definitionen in der Literatur:

> *„Telearbeit bedeutet […] elektronische Fernarbeit im Sinne der Erwerbstätigkeit mit Hilfe neuer Informations- und Kommunikationstechnologien von zu Hause oder von andernorts ‚dezentral' gelegenen Arbeitsstätten aus.“* [Kir96]
>
> Oder noch genereller: *„Telearbeit ist mediengestützte verteilte Aufgabenbewältigung.“* [RMS+00]
>
> *„The common element across all aspects of telework is the use of computers and telecommunications to change the accepted geography of work.“* [ETO04]

Um die Vielfalt der Definitionen eingrenzen und eine Arbeitsdefinition festlegen zu können, ist die Betrachtung von verschiedenen Dimensionen der Telearbeit notwendig.

Die wichtigste Dimension bei der Beschreibung von Telearbeit ist die räumliche. Die räumliche Trennung von Telearbeitsplatz und Büroarbeitsplatz ist das wesentliche Merkmal von Telearbeit. Früher wurde als Merkmalskriterium für Telearbeit häufig die Wohnung oder die Wohnortnähe des Telearbeitsplatzes verwendet. Definitionen wie

> *“Telearbeit ist [...] zu Hause oder möglichst nah vom Wohnort der Arbeitskräfte entfernt [...]“* [Hoo94] oder
>
> *“Telearbeit bezeichnet die wohnortnahe Arbeit unabhängig vom Firmenstandort [...]“* [Kor00] oder
>
> *„Der Telearbeiter nutzt die I&K-Technologien*[5] *unterstützend, um seine arbeitsvertraglichen Pflichten zumindest teilweise zu Hause oder wohnortnah zu erfüllen.“* [ReG98]

tauchen in der neueren Literatur nur noch selten auf. Dabei ist der Trend zu beobachten, dass neuere Definitionen der Telearbeit die räumliche Dimension des Arbeitsplatzstandortes weiter fassen. Dies liegt vor allem daran, dass der Ort der Arbeitsverrichtung immer unwichtiger wird, da moderne Informations- und Kommunikationstechnologien die Arbeit an jeden beliebigen Ort transportieren können [Kir96]. Auch die Bundesregierung veränderte ihre viel zitierte Definition von Telearbeit hinsichtlich des Arbeitsortes und benennt dieses Merkmal nun allgemeiner:

> *“Telearbeit ist jede [...] Tätigkeit, die [...] an einem außerhalb der zentralen Betriebsstätte liegenden Arbeitsplatz verrichtet wird [...]“* [KOW01].

Eine weitere Dimension der Begriffsbestimmung von Telearbeit stellt der zeitliche Umfang der Arbeit dar, die außerhalb der Betriebsstätte verrichtet wird. Wer permanent im Firmenbüro tätig ist, ist sicher kein Telearbeiter. Demgegenüber ist ein Arbeitnehmer, der fast seine gesamte Arbeitsleistung zu Hause und online erbringt, mit Sicherheit ein Telearbeiter. In den Publikationen zum Thema Telearbeit hat sich bezüglich der zeitlichen Dauer betriebsexternen Arbeitens das Minimum “nicht nur gelegentlich" [BMB96] oder exakter formuliert “an mindestens einem Arbeitstag pro Woche" [KOW01] etabliert. Ansonsten wäre z.B. auch ein Beschäftigter Telearbeiter, der nur einmal eine betriebliche E-Mail von zu Hause absendet, weil er es im Büro vergessen hatte. Andere Definitionen sprechen aber auch von „sporadischer oder gelegentlicher Telearbeit“ [ReG98], wenn weniger als 1/5 der Arbeitszeit zu Hause verbracht wird (siehe Kapitel 2.2.2). Auch Modelle der „supplementären Telearbeit“, also der zusätzlichen Telearbeit außerhalb der regulären Arbeitszeit, werden in der Literatur[6] erwähnt.

---

[5] I&K-Technologien – Informations- und Kommunikationstechnologien.

[6] Unter anderem in [Kor02], [JäR01].

Die letzte, aber nicht unerhebliche Dimension, die es bei der Begriffsbestimmung von Telearbeit zu berücksichtigen gilt, ist die technische. In einer Definition der Telearbeit von Rensmann und Gröpler [ReG98] wird der Zusammenhang zwischen der Arbeit und den genutzten Technologien wie folgt dargestellt:

> *„Der Telearbeiter nutzt die I&K-Technologien unterstützend, um seine arbeitsvertraglichen Verpflichtungen zumindest teilweise zu Hause oder wohnortnah zu erfüllen.“* [ReG98]

Korte [Kor00] beschreibt diesen Zusammenhang wie folgt:

> *„[...] wobei die (Zusammen-)Arbeit über räumliche Entfernungen hinweg unter primärer Nutzung von Informations- und Kommunikationstechnologien erfolgt, und eine direkte Telekommunikationsverbindung zum Arbeitgeber/Auftraggeber zur Übertragung von Arbeitsergebnissen genutzt wird.“* [Kor00]

Telearbeit liegt folglich nur dann vor, wenn elektronische Mittel und Wege zur Arbeitserbringung und zur Kommunikation mit dem Arbeitgeber oder Auftraggeber genutzt werden.

Bei der Analyse der vorhandenen Literatur zum Thema Telearbeit ergeben sich also folgende entscheidende Charakteristika bei der Definition von Telearbeit:

- **Charakteristikum “Arbeitsort“:** Der Arbeitsplatz ist räumlich vom Unternehmen getrennt.
- **Charakteristikum “Arbeitszeit“:** Die Telearbeit findet ausschließlich oder teilweise (mindestens ein Fünftel) außerhalb des Unternehmens statt.
- **Charakteristikum “Technische Infrastruktur“:** Die Tätigkeit wird mit informations- und kommunikationstechnischen Mitteln ausgeführt und die Datenübertragung zur zentralen Betriebsstätte erfolgt über elektronische Wege.

Zusammenfassend soll die folgende Definition für alle weiteren Betrachtungen eine klare Grundlage bilden:

Telearbeit

Telearbeit ist jede auf Informations- und Kommunikationstechnik gestützte Tätigkeit, die ausschließlich oder zeitweise an einem außerhalb der zentralen Betriebsstätte liegenden Arbeitsplatz verrichtet wird. Dieser Arbeitsplatz ist mit der zentralen Betriebsstätte durch elektronische Kommunikationsmittel verbunden.

Telearbeit – Leitfaden für flexibles Arbeiten in der Praxis [KOW01]

Als Problemstellung für diese Arbeit ergibt sich folglich, dass eine Lösung geschaffen werden soll, die das orts- und zeitunabhängige Arbeiten auf der Basis von modernen Informations- und Kommunikationstechnologien unterstützt.

2.2 Formen der Telearbeit

Hinter dem Begriff "Telearbeit" verbirgt sich eine Vielzahl von verschiedenen Arten der Arbeitsorganisation in Abhängigkeit von Arbeitsort, zeitlichem und rechtlichem Rahmen der Beschäftigung, technischen Möglichkeiten der Telekommunikation und der Qualifikation des Arbeitnehmers. Im folgenden sollen die in 2.1 definierten Gestaltungsmerkmale der Telearbeit dazu genutzt werden, die geläufigsten Organisationsformen der Telearbeit zu klassifizieren und ausführlich zu beschreiben, um die in ihrer Vielzahl verwirrenden Ausprägungsformen der Telearbeit für den weiteren Verlauf dieser Arbeit gegeneinander

abzugrenzen. Demnach kann Telearbeit nach folgenden Gesichtspunkten klassifiziert werden:

- räumliche Verteilung der Arbeitsplätze,
- zeitliche Einteilung der Arbeit,
- rechtliche Rahmenbedingungen.

2.2.1 Klassifizierung nach dem Arbeitsort

Als eine grundlegende Eigenschaft der Telearbeit wurde bereits die räumliche Entfernung des Arbeitsplatzes von der Betriebsstätte erwähnt. Dabei kann die Telearbeit an ganz unterschiedlichen Orten durchgeführt werden, welche da wären:

- im häuslichen Umfeld des Telearbeiters (Heimbasierte Telearbeit),
- am Standort des Kunden oder Projektpartners (On-Site-Telearbeit),
- in einem wohnortnahen Telezentrum (Center-basierte Telearbeit),
- an keinem festen Arbeitplatz, sondern mobil (Mobile Telearbeit).

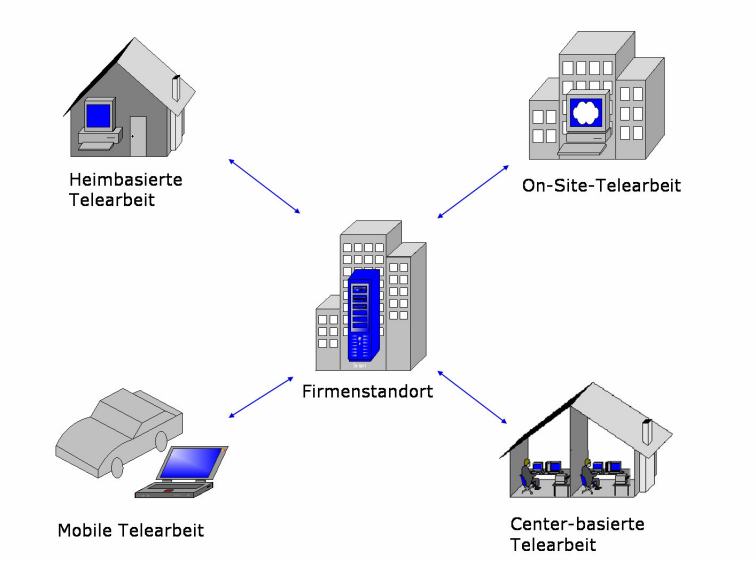

Abbildung 2-1: Formen der Telearbeit klassifiziert nach räumlicher Verteilung des Arbeitsortes

2.2.1.1 Heimbasierte Telearbeit (Home-based Telework)

Unter heimbasierter Telearbeit sind alle Formen der Telearbeit am heimischen Arbeitsplatz bzw. in der häuslichen Umgebung des Mitarbeiters zu verstehen. Dazu nutzt der Telearbeiter in seiner Privatwohnung einen mit PC und weiteren Arbeits- und Kommunikationsmitteln ausgestatteten Arbeitsplatz sowie die Anbindung über öffentliche Netze an das Firmennetz und das Internet. Die häusliche Telearbeit stellt die klassische Variante der Telearbeit dar, gemäß dem Motto „Die Arbeit kommt zum Arbeitenden, anstelle der Arbeitende zur Arbeit.“ (vgl. auch [Nil76]).

Dabei muss wiederum zwischen der Telearbeit ausschließlich zu Hause und der alternierenden Telearbeit unterschieden werden[7]. Wird die Telearbeit permanent am heimischen Arbeitsplatz ausgeführt, wird sie als "Teleheimarbeit" bezeichnet. An dieser Stelle ist jedoch darauf hinzuweisen, dass der Teleheimarbeiter, der seinen Arbeitsplatz in der eigenen Privatwohnung hat, nach dem Arbeitsrecht kein Heimarbeiter sein muss, wie der

[7] Siehe auch Kapitel 2.2.2 „Einteilung nach zeitlichem Rahmen“.

Begriff "Teleheimarbeiter" implizieren könnte. Je nach Vertrag mit dem Auftrag- oder Arbeitgeber kann der häusliche Telearbeiter die arbeitsrechtliche Stellung als Arbeitnehmer, Selbständiger/Freiberufler oder Heimarbeiter einnehmen[8]. Der Begriff Teleheimarbeiter sagt nur etwas über den Arbeitsort, die heimische Wohnung, nicht aber über die arbeitsrechtliche Form aus.

Die permanente Teleheimarbeit hat sich aber bisher kaum durchgesetzt. Bisherige Erfahrungen zeigen, dass diese Arbeitsform aufgrund mangelnder persönlicher Kontakte und der damit einhergehenden sozialen Isolation nur in Einzelfällen, z.B. bei der Integration von behinderten Menschen oder Erwerbstätigen mit eingeschränkter Mobilität, sinnvoll ist. Der Einsatz von elektronischen Kommunikationsmitteln erlaubt zwar eine komplette Verlagerung des betrieblichen Arbeitsplatzes in das häusliche Umfeld, aber es ist unrealistisch anzunehmen, dass Kommunikations- und Kooperationsbedürfnisse der Telearbeiter ausschließlich elektronisch befriedigt werden können. Deshalb wird in der Regel nicht auf eine zeitweise Präsenz im Unternehmen verzichtet. Dies gilt insbesondere für Beschäftigte, deren Aufgaben eine hohe Kommunikationsintensität voraussetzen.

2.2.1.2 On-Site-Telearbeit (On-Site Telework)

Das Telearbeiten am Standort eines Kunden, Lieferanten "[...] oder ganz allgemein am Standort des Wertschöpfungspartners [...]" [RMS+00] fällt unter die Kategorie On-Site-Telearbeit. Für zahlreiche Berufe gehört es schon heute zur alltäglichen Praxis, "vor Ort" beim Kunden oder Lieferanten zu arbeiten und dennoch über Telekommunikationsdienste mit dem eigenen Unternehmen stets in enger Verbindung zu stehen. So befinden sich die physischen Arbeitsplätze von Unternehmensberatern ebenso wie die vieler Softwareentwickler oder Systemspezialisten häufig jeweils projektbezogen am Kundenstandort [Geb02]. Auch bei dieser Form der Telearbeit steht der Telearbeiter stets mit der eigenen Firma über Telekommunikationsmedien in Verbindung. Der Arbeitsplatz des Telearbeiters befindet sich je nach Projektbedarf am Standort des jeweiligen Kunden oder Vertragpartners. Eine Besonderheit von On-Site-Telearbeit im Gegensatz zu mobiler Telearbeit ist die Tatsache, dass ein stationärer Arbeitsplatz beim Kunden oder Partner genutzt wird.

2.2.1.3 Center-basierte Telearbeit (Center-based Telework)

Bei der Center-basierten Telearbeit können vier Formen unterschieden werden:

- Satellitenbüros,
- Nachbarschaftsbüros,
- Telearbeitzentren und
- Teleservicecenter.

Satellitenbüros sind mit entsprechenden Informations- und Telekommunikationstechniken ausgestattete „Zweigstellen“ des Unternehmens außerhalb der zentralen Betriebsstätte meist in Wohnortnähe der Mitarbeiter oder Stadtrandlage. Aus der Sicht der dort arbeitenden Mitarbeiter bringen diese wohnortnahen Büros den Vorteil, dass sie ein deutlich geringeres Pendelaufkommen haben. Der Arbeitgeber hat den Vorteil, durch Satellitenbüros Arbeitskräfte für sich zu gewinnen oder an sich zu binden, die das Pendelaufkommen abschreckt oder die nicht in die Nähe der Arbeitsstätte ziehen würden.

[8] Dies gilt auch für Telearbeiter in anderen Telearbeitsformen, siehe dazu Kapitel 2.2.3 „Rechtliche Formen der Telearbeit“.

Nachbarschaftsbüros befinden sich wie Satellitenbüros in räumlicher Nähe zu den Wohnorten der Mitarbeiter. Im Gegensatz zu Satellitenbüros werden sie jedoch von mehreren Unternehmen gemeinsam genutzt und unterhalten.

Telearbeitszentren oder Telehäuser "[...] entwickeln sich vor allem in strukturschwachen Gebieten und stellen eine Telekommunikationsinfrastruktur für die lokale Wirtschaft bereit. Sie haben sich zu einer Mischung aus Serviceangebot und Telearbeitsplatz entwickelt [...]" [Kor98]. Die Büroeinheiten mit I&K-Ausstattung werden meist von einem oder mehreren privaten oder öffentlichen Trägern als Dienstleistung für mehrere Unternehmen bereitgestellt. Politische Institutionen fördern diese Form der Zentren vor allem aus arbeitsmarkt- und strukturpolitischen Gründen, um auf kommunaler und regionaler Ebene Arbeitsplätze zu schaffen. Die Idee der Telehäuser stammt aus Skandinavien. Im schwedischen Vemdalen wurde 1985 das erste Telehaus gegründet. Damit sollte auch entlegenen Regionen die Möglichkeit gegeben werden, durch die neuen I&K-Techniken dezentrale Arbeitsplätze zu schaffen.

Teleservicecenter oder Telecenter werden dagegen meist von einem Unternehmen eingerichtet und betrieben, um in diesem Zentrum Dienstleistungen, wie z.B. Sekretariats-, Telefon- oder Übersetzungsdienstleistungen auf dem freien Markt anzubieten. Das Hauptunterscheidungsmerkmal der Teleservicecenter oder Telecenter im Vergleich zu einem Satelliten- oder Nachbarschaftsbüro ist, dass externe Unternehmen die angebotenen Dienste als Kunden in Anspruch nehmen können. Entscheidend ist, dass sich in allen genannten Fällen der Arbeitsplatz des Telearbeiters in einem Telezentrum meist in Wohnortnähe befindet [Geb02] und damit das Pendelaufkommen der Arbeiter reduziert werden kann.

2.2.1.4 Mobile Telearbeit (Mobile Telework)

Als mobile Telearbeit bezeichnet man das Arbeiten an jedem beliebigen Ort außerhalb des Unternehmens bei gleichzeitiger Möglichkeit der Nutzung kommunikationstechnischer Verbindungen zu einer Zentrale für den Informationsaustausch. Der Arbeitsplatz eines mobilen Telebeschäftigten kann bspw. seine Wohnung, ein Hotel oder auch ein Auto - also jeder beliebige Ort sein. Bei dieser Form der Telearbeit handelt es sich im Wesentlichen um die bereits heute z.B. im Außen- und Kundendienst üblichen mobilen Arbeitsplätze (Laptop, Fax und Funktelefon). Diese werden jedoch zusätzlich mit entsprechenden Telekommunikationseinrichtungen (WLAN[9], Funk-Modem oder ISDN[10]-Karte) ausgestattet, so dass nicht nur die kundenorientierten Tätigkeiten sondern auch die übrigen Aufgabenbereiche ortsunabhängig erledigt werden können. Moderne mobile Netze wie z.B. UMTS[11] ermöglichen eine hohe Bandbreite und damit eine schnelle Datenübertragung sowie eine (fast) flächendeckende Verfügbarkeit.

Die hauptsächlichen Nutzenaspekte der mobilen Telearbeit liegen in der Möglichkeit eines permanenten Zugriffs auf aktuelle Informationen und Daten aus dem Unternehmen sowie der ständigen Erreichbarkeit der räumlich entfernten Mitarbeiter als auch in der Gelegenheit zum Datenaustausch mit Kunden und Geschäftspartnern.

Im Szenario mobiler Telearbeit ergeben sich spezielle Anforderungen an die Geräte- und Ortsunabhängigkeit der bereitgestellten Lösungen. Um wirklich „immer und überall“ arbeiten zu können, ist die Einbeziehung von Technologien aus dem Umfeld des

[9] WLAN - Wireless Local Area Network – drahtloses lokales Funknetzwerk.

[10] ISDN - Integrated Services Digital Network - digitales Telekommunikationsnetz.

[11] UMTS - Universal Mobile Telecommunications System - Mobilfunkstandard der International Telecommunication Union (ITU).

Mobile and Ubiquitous Computing bei der Umsetzung der technischen Unterstützung von mobiler Telearbeit dringend notwendig. Dabei ist es wichtig, dass der Mitarbeiter die bereitgestellten Dienste auf verschiedenen Endgeräten nutzen kann und zwischen diesen auch dynamisch wechseln kann.[12]

In Abbildung 2-2 sind zusammenfassend noch einmal die verschiedenen räumlichen Formen von Telearbeit dargestellt.

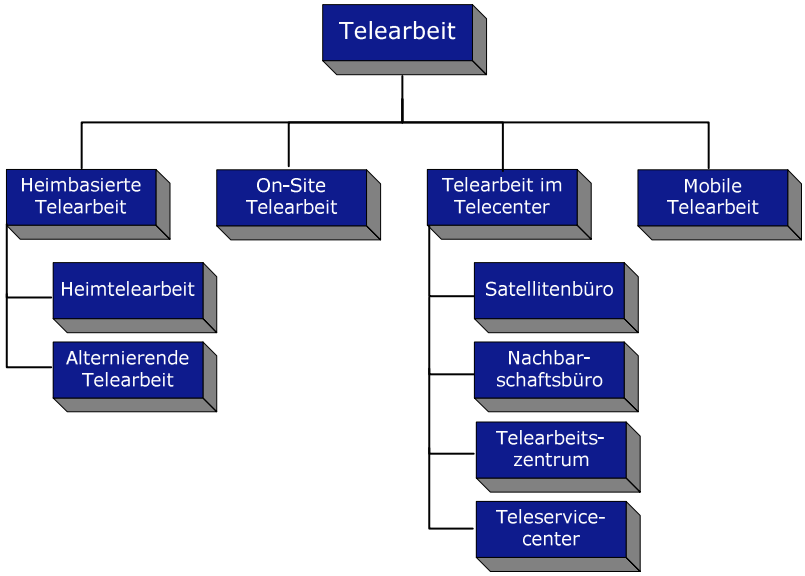

Abbildung 2-2: Räumliche Formen der Telearbeit mit ihren Unterformen[13]

2.2.2 Einteilung nach zeitlichem Rahmen

Ob ein Telearbeiter seine berufliche Tätigkeit ausschließlich an seinem Telearbeitsplatz ausführt oder ob er zeitweise seinen Arbeitsplatz zwischen dem Firmenstandort und einem außerhalb gelegenen Telearbeitsplatz wechselt, entscheidet darüber, ob er zu der Gruppe der permanenten, alternierenden oder sporadischen Telearbeiter zählt (siehe Abbildung 2-3). Dabei können auch hier die Übergänge fließend sein, da mit Telearbeit meist eine Flexibilisierung der Arbeitszeit einhergeht und die Telearbeiter je nach Bedarf entscheiden, wo sie gerade arbeiten und wie sie die Aufteilung auf die verschiedenen Arbeitsplätze gewichten. So können sie z.B. in Zeiten der Projektplanung, wenn ein großer Abstimmungsbedarf mit Kollegen notwendig wird, den meisten Teil ihrer Arbeitszeit im Büro verbringen und damit vorübergehend eher sporadisch oder gar nicht telearbeiten, und dagegen in Phasen des reinen Programmierens besser permanent zu Hause arbeiten, um ungestört zu sein. Auf die von einigen Verfassern[14] erwähnte „supplementäre“ Telearbeit, die zusätzliche Telearbeit außerhalb der regulären Arbeitszeiten, wird hier nicht explizit eingegangen, da sie dem Grunde nach zur sporadischen Telearbeit gehört.

[12] Siehe auch Kapitel 2.6.

[13] Eigene Darstellung in Anlehnung an [KOW01].

[14] Unter anderem in [JäR01], [Kor02].

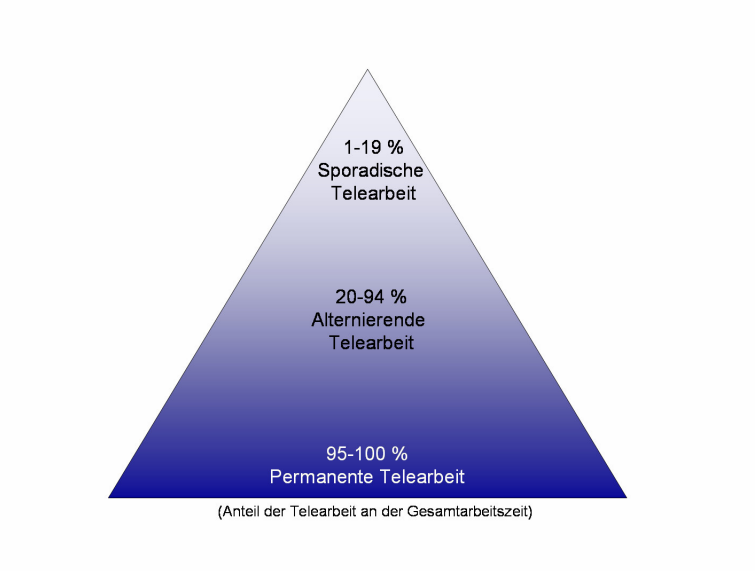

Abbildung 2-3: Zeitliche Einteilung der Telearbeit [15]

2.2.2.1 Permanente Telearbeit

Bei der permanenten oder isolierten Telearbeit erfolgt die Verrichtung der Arbeit nahezu ausschließlich außerhalb der betrieblichen Arbeitsstätte. Dabei verbringt der Telearbeiter seine gesamte Arbeitszeit an seinem häuslichen Arbeitsplatz (Teleheimarbeit) oder an einem anderen Telearbeitsplatz (z.B. im Nachbarschaftsbüro). Ein Arbeitsplatz beim Arbeitgeber steht nicht zur Verfügung.

Vor allem bei der permanenten Telearbeit zu Hause ergeben sich einige soziale Gefahren, die unter anderem in [BDH03], [Win01] und [KOW01] ausführlich beschrieben werden. Bei permanenter Telearbeit steht der Telearbeiter mit seinem Arbeitgeber und seinen Kollegen meist nur über E-Mail, elektronischen Dokumententransfer, Videokonferenzen, per Telefon oder durch andere technische Kommunikationswerkzeuge in Kontakt. Deshalb eignet sich diese Form der Telearbeit nur für solche Tätigkeiten, bei denen direkte persönliche Absprachen mit Kollegen und Vorgesetzten nicht zwingend erforderlich sind. Durch den seltenen oder fehlenden persönlichen Kontakt ist diese Form der Telearbeit nur in Einzelfällen zu empfehlen, denn einige Praxisbeispiele verzeichneten bereits eine soziale Isolation [Win01]. Hingegen erwies sich die Arbeit zu Hause als positiv bei persönlichen Belastungen des Mitarbeiters infolge von Erkrankungen innerhalb der Familie und damit verbundener häuslicher Betreuung durch den Arbeitnehmer. Ein weiterer Anwendungsfall ist die Integration behinderter Menschen mit eingeschränkter Mobilität in das Arbeitsleben, denen dadurch die Aufnahme einer Arbeit überhaupt erst möglich gemacht werden kann. Sie können in ihrem häuslichen Umfeld arbeiten, das an ihre Bedürfnisse und Einschränkungen optimal angepasst ist.

2.2.2.2 Alternierende Telearbeit

Um die genannten negativen Folgen der isolierten Telearbeit zu minimieren, findet vor allem die alternierende Telearbeit Anwendung. Auch hier kann eine klare Definition getroffen werden.

[15] eigene Darstellung in Anlehnung an [ReG98]

Alternierende Telearbeit

„Bei dieser Form der Telearbeit arbeitet der Arbeitnehmer sowohl an seinem Arbeitsplatz beim Arbeitgeber als auch in seiner Wohnung, wobei er zwischen diesen Arbeitsplätzen hin- und herwechselt."

Telearbeit – Leitfaden für flexibles Arbeiten in der Praxis [KOW01]

Ergänzend zu dieser Definition ist zu erwähnen, dass alternierende Telearbeit nicht nur bei heimbasierten Telearbeitplätzen möglich ist, sondern auch bei allen im vorangegangenen Kapitel erwähnten Telearbeitsformen. So kann z.B. auch ein Wechsel zwischen dem Büroarbeitsplatz und dem Nachbarschaftsbüro stattfinden.

Es wird empfohlen, die alternierende Telearbeit an mindestens einem und maximal drei Wochentagen auszuführen und dies auf höchstens zwei aufeinander folgende Tage zu beschränken [KOW01]. Dadurch sollen die sozialen Bindungen und der persönliche Kontakt bei der Arbeitsausübung in unverändertem Maße erhalten bleiben.

Die bei alternierender Telearbeit zusätzlich entstehenden Kosten für doppelte Arbeitsplätze können durch so genanntes „Desk-Sharing", die alternierende Nutzung des Büroarbeitsplatzes durch zwei oder mehrere Mitarbeiter, gesenkt werden. Dadurch lassen sich trotz beschränkter räumlicher Kapazitäten mehr Arbeitnehmer beschäftigen, als Arbeitsplätze vorhanden sind. Alle persönlichen Arbeitsmittel des Arbeitnehmers können während der Abwesenheit in Rollwagen untergebracht und so gegen unberechtigten Zugriff gesichert werden. Die praktische Einsetzbarkeit von Desk-Sharing kann mit einigen Beispielen aus Wirtschaft und Verwaltung belegt werden. Exemplarisch seien hier IBM Deutschland und der Landwirtschaftliche Versicherungsverein Münster (LVM) erwähnt, die seit mehreren Jahren Desk-Sharing erfolgreich einsetzen und dabei bisher vor allem positive Erfahrungen gesammelt haben.

2.2.2.3 Sporadische Telearbeit

Bei der sporadischen oder gelegentlichen Telearbeit liegt der Anteil der Telearbeit unter 1/5 der Gesamtarbeitszeit; es wird also weniger als ein Arbeitstag pro Woche am Telearbeitsplatz verbracht. Deshalb wird sie in vielen Veröffentlichungen gar nicht erwähnt, soll hier aber der Vollständigkeit halber genannt werden. Die Einrichtung eines Telearbeitsplatzes inklusive der notwendigen Geräte und Installationen erfordert erhebliche Investitionen, die bei sporadischer Telearbeit meist nicht gerechtfertigt sind. Ausnahmen bilden Telearbeitsplätze, die vom Arbeitnehmer selbst zur Verfügung gestellt werden, und oft zur Erledigung zusätzlicher Aufgaben bzw. für Überstunden (supplementäre Telearbeit) oder für einen langsamen Einstieg in die Telearbeit genutzt werden.

2.2.3 Rechtliche Formen

Telearbeit kann in unterschiedlichen Rechtsverhältnissen erbracht werden, die hier nur kurz skizziert werden sollen. Vertiefende Ausführungen zu rechtlichen Aspekten der Telearbeit finden sich unter anderem in [Col95], [KOW01], [Wed94] und [Joh97].

Telearbeit ist denkbar in Form

- des Arbeitsverhältnisses,
- der Heimarbeit oder
- der selbständigen freiberuflichen Tätigkeit.

Bei der Wahl der Vertragsgestaltung sollten die Bedürfnisse von allen Vertragspartnern berücksichtigt werden. Dabei ist zu beachten, dass der rechtliche Status nicht nur von der vertraglichen Gestaltung abhängt, sondern auch von der tatsächlichen Arbeitsumgebung [KOW01]. Telearbeit steht dabei im Spannungsfeld von sozialer Sicherheit auf der einen und Flexibilität und Selbstständigkeit der Telearbeiter auf der anderen Seite [Col95]. Grundsätzlich lässt sich bei den verschiedenen Formen der Arbeitsverhältnisse eine Abhängigkeit der sozialen Sicherheit vom Grad der Selbständigkeit der Erwerbstätigen feststellen. (siehe Abbildung 2-4).

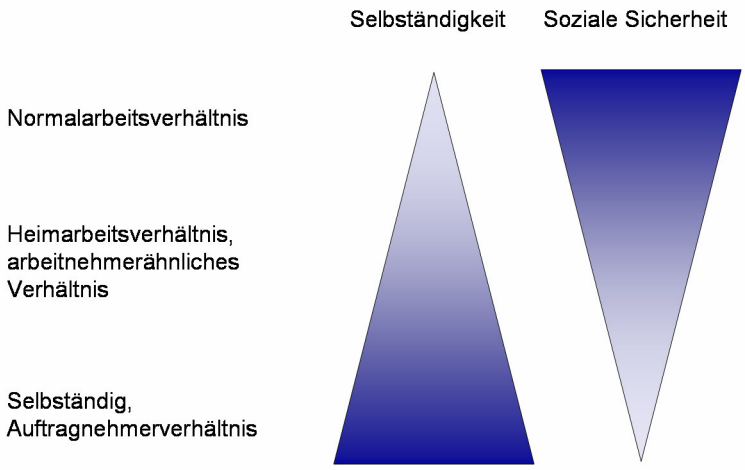

Abbildung 2-4: Soziale Aspekte des Arbeitsverhältnisses[16]

2.2.3.1 Arbeitnehmer

Die häufigste Rechtsform der Telearbeit ist die Beibehaltung des Arbeitnehmerstatus [Wed94]. Für eine Arbeitnehmerstellung des Telearbeiters spricht einerseits die bessere soziale Absicherung und anderseits seine Weisungsgebundenheit gegenüber seinem Arbeitgeber. Diese wirkt sich darin aus, dass der Arbeitgeber Art, Dauer, Umfang der Tätigkeit bestimmt und entsprechende Rechte an der Arbeitsleistung und dem Arbeitsergebnis hat. Soweit Telearbeit in der Form des Normalarbeitsverhältnisses erbracht wird, findet das Arbeits- und Sozialrecht, welches in der Regel die bessere soziale Absicherung vorsieht, grundsätzlich Anwendung [Col95].

Der Arbeitnehmerstatus wird in der Rechtsprechung unter anderem nach folgenden Kriterien festgelegt:

- Fremdnützigkeit der Arbeit,
- Weisungsgebundenheit des Arbeitnehmers und Direktionsrecht des Arbeitgebers,
- Eingliederung in den Betrieb und die betriebliche Organisation,
- Bestimmung von Arbeitsort und -zeit durch den Arbeitgeber.

Da es jedoch bisher kein „Telearbeitsgesetz“ mit eigenständigen gesetzlichen Regelungen gibt, sind Sonderregelungen dergestalt nötig, dass einige Punkte entweder durch Tarifver-

[16] eigene Darstellung in Anlehnung an [Col95]

träge, Betriebsvereinbarungen oder individuelle Vereinbarungen zwischen Telearbeiter und Arbeitgeber geklärt werden müssen [Wed94]. Dazu zählen folgende Punkte:

- Freiwilligkeit der Teilnahme an der Telearbeit und Rückkehrmöglichkeiten an den betrieblichen Arbeitsplatz,
- Arbeitszeitfestlegungen, Vergütung und Zuschläge für Mehrarbeit,
- Bereitstellung der Arbeitsmittel, Nutzungsvereinbarungen,
- Aufwendungen für Fahrten zwischen Betrieb und Telearbeitsplatz,
- Aufwendungen für Nutzung des Wohnraums, Strom und Heizung,
- Haftung (bei Diebstahl oder Beschädigung der IT, aber auch bei Arbeitsunfall oder Berufskrankheit),
- arbeits- und datenschutzrechtliche Anforderungen an den Telearbeitsplatz,
- Befugnisse der Arbeitgeber und Arbeitnehmervertretungen, Zugangsrecht zur privaten Wohnung, Persönlichkeitsschutz der Arbeitnehmer.

Vorreiter bei der vertraglichen Vereinbarung von Telearbeit in Deutschland war die IBM Deutschland GmbH. Bereits 1991 schloss die Unternehmensführung mit dem Gesamtbetriebsrat eine Betriebsvereinbarung über „außerbetriebliche Arbeitsstätten“ ab [ReG98]. Anfänglich war die Rechtsunsicherheit ein großes Hemmnis bei der Einführung von Telearbeit in Deutschland. Mittlerweile existieren aber eine Reihe von Musterverträgen, die vor allem durch die Gewerkschaften und Betriebsräte erstellt wurden.

2.2.3.2 Heimarbeiter

Telearbeit ist auch in der Form der Heimarbeit im Sinne des Heimarbeitsgesetzes durchführbar. Da Heimarbeiter lediglich wirtschaftlich, nicht aber persönlich abhängig sind, werden sie nicht als Arbeitnehmer definiert. Sie können Dauer der Tätigkeit und Lage des Arbeitsplatzes frei bestimmen und unterliegen daher nicht dem so genannten Direktionsrecht eines Arbeitgebers. Aus diesen Gründen findet das Arbeitsrecht auf sie keine Anwendung, dennoch ist das Arbeitsgericht für Streitigkeiten zwischen Auftraggeber und Heimarbeiter zuständig. Rechtsgrundlagen sind das Heimarbeitsgesetz (HAG) und entsprechende bindende Festsetzungen zu Arbeitszeitschutz, Gefahrenschutz, Entgeltregelung sowie Kündigungsschutz. Wegen ihrer Geltung als Beschäftigte sind Heimarbeiter voll sozialversicherungspflichtig [Joh97].

2.2.3.3 Selbständige

Soweit Telearbeit von Selbständigen erbracht wird, bestimmen sich die Rechtsbeziehungen zwischen Auftraggeber und Telearbeiter nicht nach dem Arbeitsrecht, sondern nach den Vorschriften über den privatrechtlichen Werkvertrag. Der Telearbeiter kann Art, Dauer und Umfang seiner Tätigkeit selbst bestimmen und unterliegt keinen Weisungen einer vorgesetzten Person. Dabei ist er für eine oder mehrere Firmen tätig, wobei er für sich und ggf. andere das unternehmerische Risiko trägt [Joh97].

2.2.4 Fazit

Zusammenfassend kann man sagen, dass der heutige Stand der Informations- und Kommunikationstechnologien eine Verlagerung des Arbeitsplatzes an jeden beliebigen Ort außerhalb des Unternehmens ermöglicht und damit der Begriff „Arbeitsplatz“ eine ganz neue Dimension erlangt. Mit der Möglichkeit der flexiblen Ortswahl verschwimmen langsam auch die starren Grenzen zwischen den einzelnen Formen immer mehr. Es kann Beschäftigte geben, die sowohl zu Hause als auch unterwegs oder beim Kunden arbeiten.

Die verschiedenen Arten der Telearbeit sind ganz unterschiedlich verbreitet und akzeptiert. Durch die Vielzahl von verschiedenen Definitionen und die fließenden Übergänge zwischen den Formen sind quantitative Einschätzungen sehr schwierig und kaum vergleichbar. Deshalb sollen hier nur kurz einzelne Untersuchungen erwähnt werden. In [Kor02] wurden die Ergebnisse einer Studie zur Verbreitung der verschiedenen Formen von Telearbeit in der EU veröffentlicht. Kordey unterscheidet hier drei Grundformen: häusliche Telearbeiter, mobile Telearbeiter und selbständige Telearbeiter in so genannten SOHOs[17]. Die genannten drei Grundformen der Telearbeit sind aber nicht überschneidungsfrei; sowohl häusliche Telearbeiter als auch Selbständige in SOHOs können gleichzeitig mobile Telearbeit betreiben. Wie nachfolgende Grafik (Abbildung 2-5) verdeutlicht, arbeiten weit mehr als die Hälfte (57%) der in der EU vorgefundenen Telearbeiter in Form häuslicher Telearbeit, weitere 31% sind als mobile Telearbeiter tätig und ca. ein Viertel (26%) können der Kategorie der selbständigen Telearbeiter zugeordnet werden.

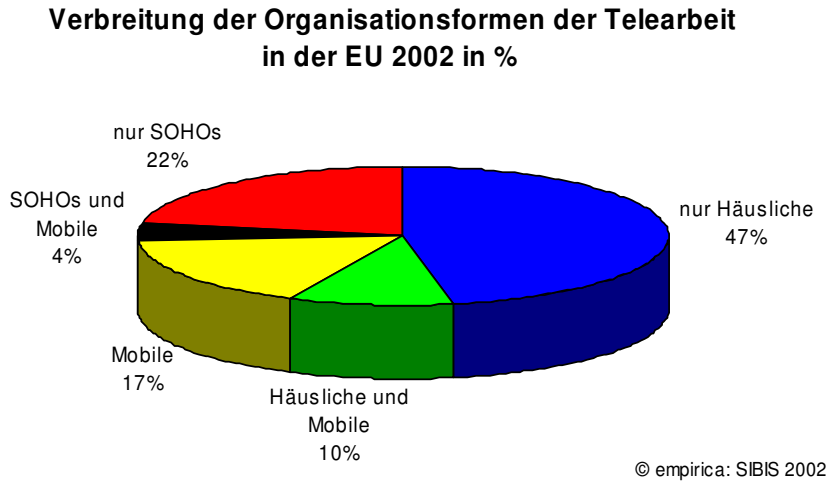

Abbildung 2-5: Verbreitung der Telearbeitsformen in der EU [Kor02]

Was die zeitliche Aufteilung der Arbeit betrifft, ist alternierende Telearbeit sicherlich die häufigste Form der Telearbeit. Von den, nach einer Erhebung des Fraunhofer Instituts für Arbeitswirtschaft und Organisation [IAO97], 875.000 Telearbeitern in Deutschland waren 350.000, mithin 40 % in alternierender Telearbeit tätig. Von einer weitergehend hohen Verbreitung der alternierenden Telearbeit kann auch zukünftig ausgegangen werden, zumal in privatwirtschaftlichen Unternehmen sowie in öffentlichen Verwaltungen bei der Durchführung von Telearbeit überwiegend das Modell der alternierenden Telearbeit gewählt wird.

Hauptgrund für die höhere Akzeptanz der alternierenden Telearbeit ist die flexible Gestaltung der Arbeitszeiten und -bedingungen. Dies bewirkt eine gesteigerte Mitarbeitermotivation und damit höhere Produktivität des Einzelnen. Verschiedene Studien (Hewlett Packard 1995, IBM USA 1995 [BMB96]) gehen davon aus, dass ein Telearbeiter 10-25 % produktiver arbeitet als ein normaler Arbeitnehmer.[18]

Der zeitliche Rahmen wird durch den Arbeitsinhalt und die Eigenverantwortlichkeit und Qualifikation des Arbeiters bestimmt. Der negative Einfluss isolierter Telearbeit auf das

[17] SOHO (small office, home office) beschreibt Telearbeit von Selbständigen, deren hauptsächlicher Arbeitsort das häusliche Kleinstbüro ist bzw. die das Heimbüro als Ausgangspunkt für ihre Arbeit an unterschiedlichen Orten nutzen.

[18] Weitere Ausführungen zu den Vorteilen von Telearbeit in Kapitel 2.5.

soziale Gefüge der Mitarbeiter wird durch den Wechsel zwischen innerbetrieblicher und Telearbeit unterbunden.

2.3 Anwendungsszenarien

Die primäre Idee bei der Einführung von Telearbeit ist, dass die Arbeit zum Arbeitenden anstatt der Arbeitende zur Arbeit transportiert wird [BSZ01]. Für eine Verlagerung der Beschäftigung vom zentralen Arbeitsort zum Wohnort oder einem anderen Ort außerhalb der Betriebsstätte eignen sich vor allem Tätigkeiten zur Erstellung, Bearbeitung und Weitergabe von Informationen, die einen hohen Anteil selbständiger Arbeit zulassen und eine Präsenz im Unternehmen nicht dringend voraussetzen. Um das weite Feld möglicher Anwendungen der Telearbeit zu gliedern, wurden verschiedene Kriterien zur Klassifikation von Teleworking-Szenarien heraus gearbeitet.

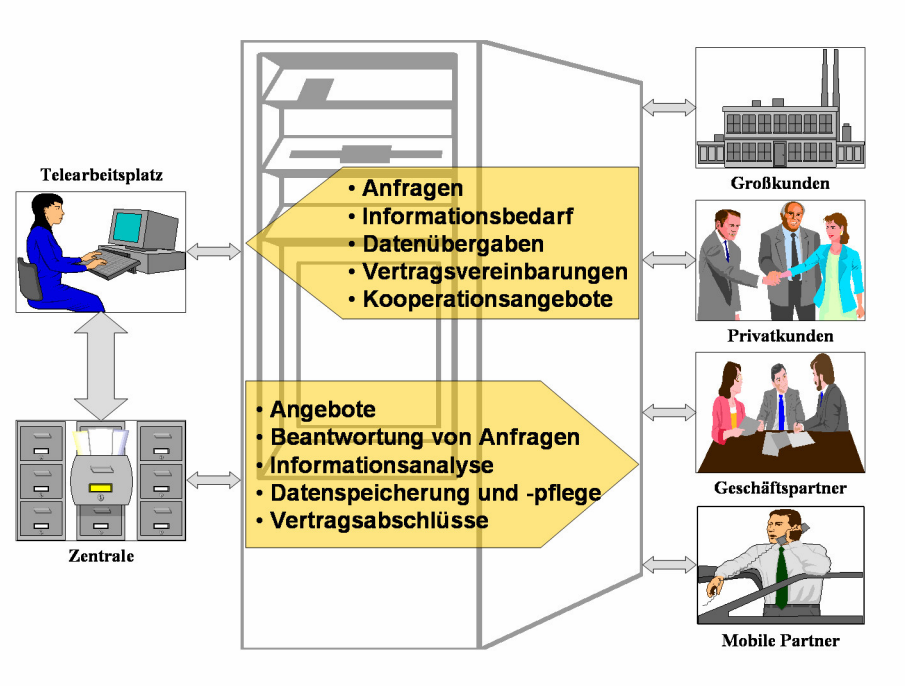

Abbildung 2-6: Einsatzszenario für Telearbeit in einem Finanzdienstleistungsunternehmen

Abbildung 2-6 zeigt ein konkretes Anwendungsszenario, das für ein Finanzdienstleistungsbüro in Sachsen geplant und als Pilotlösung im Rahmen des Landesinnovationskollegs „Anwendungsnahe Teledienste für Industrie und Verwaltung in Sachsen“ umgesetzt wurde. Die Telearbeiter kommunizieren über verschiedene Internetdienste sowohl mit der eigenen Unternehmenszentrale auf der einen als auch mit Kunden und Geschäftspartnern auf der anderen Seite. Um Anfragen und Angebote der Partner bearbeiten zu können, benötigt der Telearbeiter Zugriff auf alle relevanten Informationen und Daten in der Unternehmenszentrale. Als Ergebnis sendet der Telearbeiter Antworten auf Anfragen, Angebote, Vertragsabschlüsse etc. direkt an die Kunden oder Partner. Dabei sollte es für die Kunden und Partner völlig transparent sein, ob sie mit einem Telearbeiter oder mit einem Angestellten in der Zentrale kommunizieren.

2.3.1 Für Telearbeit geeignete Tätigkeiten

Die oft gestellte Frage nach den geeigneten Arbeitsplätzen ist nicht einfach zu beantworten. Was möglich ist, hängt stark von der Bereitschaft zur Reorganisation der Arbeitsabläufe und dem Willen der Beteiligten ab. Das größte Hindernis ist oft die Zurückhaltung der Arbeitgeber, vor allem des mittleren Managements.

Dennoch lassen sich Tätigkeiten identifizieren, die sich besonders gut, d.h. ohne größere Veränderung der Abläufe, für Telearbeit eignen. Diese müssen folgende Anforderungen erfüllen:

- hoher Anteil an Informationsverarbeitung,
- Phasen längerer, konzentrierter Arbeit,
- definierte, überprüfbare Ergebnisse,
- zeitlich planbarer Zugriff auf nicht digitalisierte Aktenbestände,
- geringer Bedarf an nichtelektronischen Geräten,
- geringer Bedarf an Face-to-Face-Kommunikation,
- geringer Platzbedarf.

Die wichtigste Anforderung ist die physikalische Auslagerungsfähigkeit der Arbeit, d.h. die notwendigen Arbeitsmittel müssen durch einen Online-Zugriff verfügbar gemacht werden können [KOW01]. Die allgemeinen Anforderungen werden zusätzlich durch spezielle, für die Tätigkeit notwendige Anforderungen ergänzt.

Mit dem in den letzten Jahren vollzogenen Wandel der Organisationsstrukturen der Unternehmen und dem rasanten Fortschritt der Technik hat sich auch die Einschätzung geändert, welche Arbeiten dezentral durchgeführt werden können. Während früher vorwiegend gering qualifizierte und leicht kontrollierbare Routinetätigkeiten wie Daten- und Texterfassung im Vordergrund standen, hat sich der Schwerpunkt in den Bereich qualifizierterer Arbeiten verschoben. Deshalb wird im Folgenden zwischen Unterstützungstätigkeiten und höher qualifizierten Tätigkeiten unterschieden. Die Unterstützungstätigkeiten umfassen alle Tätigkeiten, deren Arbeitsleistungen quantitativ messbar sind und eine hochgradige Unabhängigkeit bei der Aufgabenerfüllung gewähren. Die Aufgabe muss folglich zu erfüllen sein, ohne ständig auf neue Instruktionen und Ergebnisse anderer Mitarbeiter angewiesen zu sein. Einige derartige Tätigkeiten sind in der folgenden Übersicht (Tabelle 2-1) zusammengefasst.

Daten- und Texterfassung	Textverarbeitung	Vorlagen- und Satzerstellung
Auftragsannahme	Hot-Line Service	Reservierungsdienste
PR-Tätigkeiten	Tele-Marketing	Statistik
Übersetzungen	Dokumentation	Vorbereitung von Lehrmaterialien
Informationsrecherche	Information Broker	Buchhaltung

Tabelle 2-1: Unterstützungstätigkeiten bei Telearbeit (in Anlehnung an [KOW01])

Bei den höher qualifizierten Tätigkeiten steht die weitestgehend selbständige Erledigung und Planung der Arbeitsaufgaben auf der einen Seite einer Abhängigkeit zwischen verschiedenen Aufgaben und von Zuarbeiten durch andere Mitarbeiter auf der anderen Seite gegenüber. Damit höher qualifizierte Tätigkeiten für die Telearbeit geeignet sind, müssen sie strukturierbar und Meilensteine festlegbar sein, da sich die Arbeitsleistung sonst nur schwer bewerten lässt. Es wird in diesem Zusammenhang von einer ergebnisorientierten

Leistungsbewertung gesprochen, die sich in dem Managementprinzip „*Management by Objectives*“ widerspiegelt. Während bei den Unterstützungstätigkeiten vor allem quantitative Ergebnisse bewertet werden, sind es bei den höher qualifizierten Tätigkeiten vor allem qualitative Aspekte, die den Erfolg der Arbeit bestimmen. Unter dieser Voraussetzung muss die Koordination der unterschiedlichen Aufgaben und beteiligten Personen durch eine Telearbeitsumgebung geeignet unterstützt werden. Die folgende Übersicht (Tabelle 2-2) zeigt eine grobe Einordnung verschiedener höher qualifizierter Tätigkeiten, die in Telearbeit realisierbar sind.

Finanzdienstleistungen	▪ Controlling ▪ Kalkulation ▪ Auftragsbearbeitung
Beratungstätigkeiten	▪ Kundenberatung und Consulting ▪ Finanzberatung ▪ Steuerberatung ▪ DV-Beratung ▪ Juristische Beratung
Support	▪ Kunden- und Außendienst ▪ DV- und Fernwartung ▪ Systemanalyse
Redaktionelle Tätigkeiten	▪ Autorentätigkeiten ▪ Forschungstätigkeiten ▪ Journalistische und Redakteurstätigkeiten ▪ Erstellung juristischer Dokumente und Schriften ▪ Gutachtertätigkeiten
Kreative und produktive Tätigkeiten	▪ Planung und Konstruktion ▪ Programmierung ▪ Datenbankentwicklung ▪ Grafik und Design ▪ Architektentätigkeiten ▪ Technisches Zeichnen, CAD ▪ Produktgestaltung ▪ Erstellung von Lehrmaterialien

Tabelle 2-2: Höher qualifizierte Tätigkeiten bei Telearbeit (in Anlehnung an [KOW01])

Zusammenfassend kann man feststellen, dass folglich alle Tätigkeiten als Anwendungsfelder von Telearbeit in Frage kommen, deren Arbeitsaufgaben mittels Informations- und Telekommunikationstechniken ortsunabhängig erfüllt werden können. Insbesondere eignen sich informationsverarbeitende (wie Programmierer, Softwareentwickler) und redaktionelle (wie Autoren, Lektoren, Gutachter) Berufe mit klar abgegrenzten Arbeitsergebnissen für Telearbeit. Abbildung 2-7 zeigt die Ergebnisse einer Studie der TA Telearbeit [BMB96], in der 78 Unternehmen zu den Tätigkeitsfeldern ihrer Telearbeiter befragt wurden. Dabei wird deutlich, dass einfache Tätigkeiten wie Textverarbeitung ebenso wie höher qualifizierte Fachaufgaben und Management-Arbeiten für Telearbeit geeignet sind.

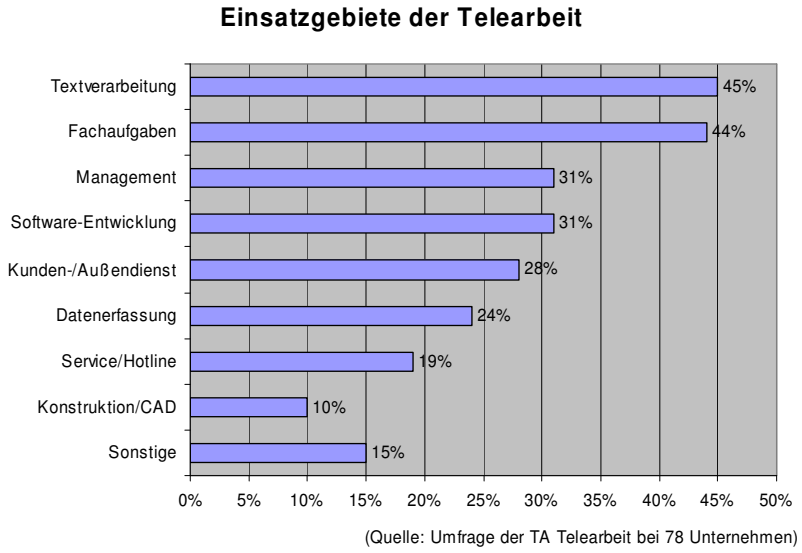

Abbildung 2-7: Einsatzgebiete der Telearbeit [BMB96]

2.3.2 Auswahl der Telearbeiter

Die Auswahl der richtigen Mitarbeiter ist der Schlüssel zu einer erfolgreichen Einführung der Telearbeit. Erster Gesichtspunkt ist die Freiwilligkeit, die auch das Recht auf die Rückkehr auf den betrieblichen Arbeitsplatz einschließt. Als Auswahlkriterien sind Eigenmotivation, Selbstdisziplin, Flexibilität, Selbständigkeit und Fachkenntnisse gebräuchlich. Diese Eigenschaften lassen sich oft nur bei längerer Betriebszugehörigkeit sicher beurteilen. Darüber hinaus müssen die häuslichen Verhältnisse ungestörtes Arbeiten zulassen. Telearbeiter sollten außerdem in der Lage sein, kleinere Störungen ihrer technischen Ausrüstung selbst zu beheben. Weitere Anforderungen bei der Auswahl der Telearbeiter haben Kordey und Korte in [KoK98] wie folgt (siehe Tabelle 2-3) kategorisiert:

Eignung der Aufgabe
- Anteil der notwendigen Face-to-Face-Kontakte, Meetings,
- Grad der Zusammenarbeit mit Kollegen, Untergebenen oder Vorgesetzten,
- Notwendigkeit des Zugriffs auf (nicht digitalisierte) Unterlagen, Produkte etc.,
- Anteile nicht bzw. schwer planbarer Ad-hoc-Aufgaben,
- Anteile von Aufgaben, die ungestörtes Arbeiten erfordern.

Eignung der Person
- fachliches Können, Eigenständigkeit,
- Dauer der Firmenzugehörigkeit, Berufserfahrung, Vertrauen,
- Selbstdisziplin und Eigenmotivation,
- technisches Verständnis, Innovationsbereitschaft.

Eignung der häuslichen und familiären Umstände
- Arbeitszimmer,
- Ablenkungs- und Störungspotenzial (Familienmitglieder, Nachbarn).

Dringlichkeit bzw. soziale Umstände
- zu betreuende Kinder bzw. pflegebedürftige Angehörige,
- Länge und Zeitdauer des Pendelweges.

Tabelle 2-3: Kriterien für die Eignung von Telearbeitern (in Anlehnung an [KoK98])

Das wichtigste Kriterium bei der Auswahl ist aber sicher das notwendige Vertrauensverhältnis zwischen Telearbeiter und Vorgesetztem. Ohne eine jahrelange Zusammenarbeit ist dies meist noch nicht vorhanden. Deshalb sind Pläne, mit Telearbeit neue Arbeitsplätze zu schaffen, bisher in der Praxis auch immer gescheitert.

2.4 Praktische Umsetzung von Telearbeit

Bei der bisherigen technischen Ausstattung von Telearbeitsplätzen gibt es große Unterschiede. Zu vielfältig sind die individuellen Berufsinhalte und Arbeitsformen von Telearbeitern, zu unterschiedlich die technischen Ansprüche. Bestimmte Komponenten gehören aber zu jedem Telearbeitsplatz: Hardware, Software und Netzzugang. Für die schnelle und effektive Arbeit im Telebüro sind heute alle technischen Voraussetzungen geschaffen. Exakte Planung ist dennoch wichtig, denn die verwendeten Geräte und Software sind maßgeblich für die Qualität und die Effektivität der geleisteten Arbeit verantwortlich. Dabei müssen vor allem auch die tätigkeitsbezogenen Anforderungen und Bedürfnisse des Telearbeiters berücksichtigt werden.

Unter diesen Gesichtspunkten sollen nun die drei Hauptkomponenten Hardware, Software und Netztechnologie genauer betrachtet werden.

2.4.1 Hardware

Unter Hardware versteht man die gängigen Komponenten wie PC mit Monitor, Maus und Tastatur. Alternativ kann bei den derzeitigen Preisen für Laptops sicher ein derartiges Geräte angeschafft werden. Dies würde vor allem Telearbeitern entgegenkommen, die in beengten Wohnverhältnissen leben oder mobile Telearbeit praktizieren. Je nach Anforderung sind diese Komponenten durch Drucker, Modem/ISDN/WLAN, Telefon, Telefax, Anrufbeantworter usw. zu ergänzen. Die Anschaffung kann aber nicht ohne vorherige Festlegung der später verwendeten Software und Netztechnologien erfolgen. Sollte z.B. die Netzanbindung sehr leistungsfähig sein, könnten anstelle des Telefons Technologien wie Voice-Over-IP zur Kommunikation genutzt werden. Dies setzt aber einen leistungsfähigen Rechner voraus. Ebenso könnte die Abwicklung von Faxen über den Rechner geschehen und somit wäre die Anschaffung eines separaten Faxgerätes überflüssig.

Generell sollte das Hauptaugenmerk auf den vier Komponenten Prozessortaktrate, Hauptspeichergröße, Festplattengröße und Erweiterungsfähigkeit liegen. Die entsprechenden Anforderungen an diese Komponenten richten sich dann vor allem nach den verwendeten Programmen. Sollen Spezialanwendungen wie CAD- oder Grafikprogramme zum Einsatz kommen, so ist ein leistungsfähigeres Gerät vonnöten. Außerdem sollte, in Hinblick auf die Investitionssicherheit, bei der Anschaffung der Geräte auf ausreichende Reserven geachtet werden.

Eine Befragung von 20 Telearbeitern im Rahmen des Projektes „intermobil“ (1999-2003) zur Ausstattung ihres Telearbeitsplatzes ergab folgende in Abbildung 2-8 dargestellte Verteilung der zur Verfügung stehenden Rechentechnik und Kommunikationsanschlüsse. Die dabei gewonnenen Aussagen kann man aber aufgrund der geringen Probandenzahl und der rasanten Entwicklung auf dem IT-Markt nicht verallgemeinern, sie können aber einen Anhaltspunkt für eine grobe Bewertung geben. Bei der Auswertung kann man eine deutliche Mehrheit von ISDN-Anschlüssen erkennen, die zum Teil in Verbindung mit ADSL genutzt werden. Die Nutzung des Mobilfunks beschränkt sich bisher eher auf den Bereich des Telefonierens und nicht auf den Datentransfer. Der Trend in den nächsten Jahren geht auf jeden Fall zu schnelleren Datenverbindungen wie ADSL und der Nutzung von Mobilfunknetzen wie UMTS für den Datentransfer.

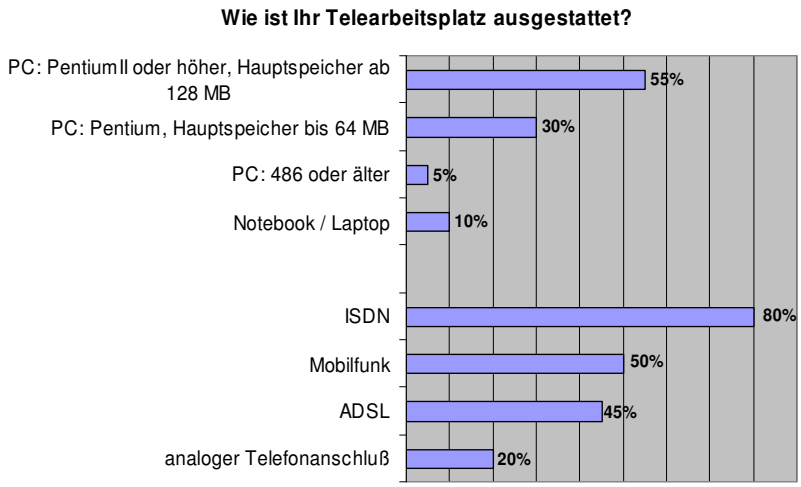

Abbildung 2-8: Ausstattung des Telearbeitsplatzes (Befragung intermobil 2003)

Erstaunlich ist die geringe Verbreitung von Laptops bei nur 10 % der Befragten. Die Entwicklung immer leistungsfähigerer und preiswerterer Notebooks in den letzten Jahren hat sicher zu einer deutlichen Steigerung des Anteils von Laptops bei Telearbeitern geführt, da sie ein flexibles ortsunabhängiges Arbeiten deutlich besser unterstützen als Desktop-PCs. Auch lassen sich hier Einsparungspotenziale erzielen, wenn der Laptop auch als Arbeitsgerät im Büro genutzt wird. Die oft als Hinderungsgrund für Telearbeit genannten Kosten für einen zusätzlichen Telearbeitsrechner entfallen dadurch.

2.4.2 Netztechnologien

Genau wie bei der Wahl der notwendigen Hardware sollten auch bei der Auswahl der Netzanbindung die speziellen Erfordernisse der Arbeitsaufgabe berücksichtigt werden. Überdimensionierte Anbindungen verursachen nur unnötige Kosten, wohingegen unterdimensionierte Anbindungen ein verzögerungsfreies Arbeiten unmöglich machen.

2.4.2.1 Analoger Datentransfer

Mittels Modem und entsprechender Software kann die normale Telefon- bzw. Sprachleitung zur Datenübertragung genutzt werden. Dabei sind Übertragungsraten von 56 kbit/s möglich, was für normalen E-Mail-Verkehr bzw. für kleine zu übertragende Dateien und für das Betrachten der meisten Internetseiten durchaus ausreichend ist. Nachteil dieser Anbindung ans Datennetz ist die Nutzung der Sprachleitung, was die parallele Nutzung des Telefons während der Datenübertragung nicht erlaubt. Aus diesen Gründen werden analoge Modems nur noch sehr selten für Telearbeit verwendet, sollten aber der Vollständigkeit halber Erwähnung finden.

2.4.2.2 Integrated Services Digital Network (ISDN)

ISDN bezeichnet einen internationalen Standard für den Zugang in ein digitales Telekommunikationsnetz, das hauptsächlich für die Übertragung von Telefongesprächen genutzt wird. Im Zuge des Internet-Booms wurde ISDN auch zunehmend für die Datenübertragung verwendet, da es verglichen mit der analogen Datenübertragung per Modem schneller und somit auch kostengünstiger ist. Diese europaweit standardisierte digitale Verbindung ermöglicht es, auf zwei digitalen Nutzkanälen mit jeweils 64 kbit/s sowohl Sprache als auch Nutzdaten zu übertragen. Durch das Vorhandensein zweier Kanäle ist das gleich-

zeitige Telefonieren und Arbeiten im Netz möglich. Es besteht zusätzlich die Möglichkeit, beide Kanäle zu bündeln und so eine Datenrate von 128 kbit/s zu realisieren. Damit sind im eingeschränkten Maße Bildtelefonie und Videokonferenzen möglich. Sollten noch höhere Datenraten nötig sein, besteht die Möglichkeit, durch den Primärmultiplexanschluss bis zu 30 Verbindungen mit jeweils 64 kbit/s parallel zu nutzen. Bei dieser Kanalbündelung entstehen Übertragungskapazitäten bis zu 1,92 Mbit/s.

2.4.2.3 Asymmetric Digital Subscriber Line (ADSL)

Mittlerweile wird aufgrund höherer Bandbreite und geringerer Kosten zunehmend *Digital Subscriber Line* (DSL) für den Zugang zum Internet eingesetzt; eine Technik, die eine vergleichsweise breitbandige digitale Verbindung ermöglicht, die erheblich schneller ist als ISDN. Mit dieser Technologie können auf der normalen Kupferleitung Datenraten von bis zu 8 Mbit/s zum normalen Telefonanschluss realisiert werden. In der Gegenrichtung ist ein Datenstrom von bis zu 768 kbit/s möglich. Dieser Unterschied bzgl. der Datenraten gibt dieser Form des DSL seinen Namen – asymmetrisch. Der Grund für die geringere Datenrate vom Anschluss ins Netz ist der Tatsache geschuldet, dass üblicherweise mehr Daten empfangen als gesendet werden. Eine weitere Form des DSL ist *High data rate Digital Subscribe Line* (HDSL), die aber bisher noch keine breite Anwendung im Privatkundenbereich findet. Mit HDSL werden symmetrische Bitraten übertragen, also in beiden Richtungen die gleiche Bandbreite. Bei beiden genannten Arten des DSL kann parallel zum Datenverkehr telefoniert werden, was eine uneingeschränkte Erreichbarkeit des Telearbeiters gewährleistet.

2.4.2.4 Mobilfunknetze

Mobile Netze gewinnen in den letzten Jahren unter dem Paradigma „*Work where you are*“ vor allem im Bereich des mobilen Telearbeitens immer mehr an Bedeutung. Waren Datenverbindungen über mobile Netze früher äußerst teuer und nur sehr schmalbandig, haben sich in den letzten Jahren Technologien durchgesetzt, die eine größere Übertragungsbandbreite zu einem wesentlich günstigeren Preis erlauben. Zu diesen Technologien gehört neben GSM[19] und GPRS[20] auch UMTS[21]. Das weltweit erste UMTS-Netz wurde 2001 durch die Manx Telecom auf der Isle of Man in Betrieb genommen. Die österreichische Mobilkom Austria startete am 25.September 2002 das erste nationale UMTS-Netz Europas und seit 2004 ist UMTS auch in Deutschland kommerziell verfügbar. Derzeit bauen die deutschen UMTS-Netzbetreiber ihre Netze im FDD-Modus auf; die damit erzielbare Datentransferrate liegt derzeit bei 384 kbit/s für den Downlink [WIK05].

Aber auch drahtlose lokale Netze wie WLAN[22] und Bluetooth[23] spielen eine immer größere Rolle für flexible Telearbeit. *Bluetooth* bietet eine drahtlose Schnittstelle, über die sowohl mobile Kleingeräte wie Mobiltelefone und PDAs als auch Computer und Periphe-

---

[19] GSM - Global System for Mobile Communications: weltweiter Standard der ETSI (European Telecommunications Standards Institute) für digitale, zellulare Mobilfunknetze.

[20] GPRS - General Packet Radio Service: Erweiterung des GSM-Mobilfunk-Standards um paketorientierte Datenübertragung.

[21] UMTS - Universal Mobile Telecommunications System: Mobilfunkstandard der ITU (International Telecommunication Union).

[22] WLAN - Wireless Local Area Network: Bezeichnung für drahtlose lokale Funknetzwerke (IEEE-Standard 802.11).

[23] Bluetooth: Industriestandard für die drahtlose Vernetzung von Geräten über kurze Distanzen (IEEE Standard 802.15).

riegeräte miteinander kommunizieren können. Ein solches Netzwerk wird auch als *Wireless Personal Area Network* (WPAN) bezeichnet, da es nur eine geringe Reichweite hat.

Im Gegensatz zu WPAN haben WLAN größere Sendeleistungen und Reichweiten und bieten im Allgemeinen höhere Datenübertragungsraten. Ein WLAN kann man in zwei Modi betreiben:

- Im Infrastruktur-Modus wird eine Basisstation, ein so genannter *Access Point*, speziell ausgezeichnet. Er koordiniert die einzelnen Netzknoten. Häufig ist dieser Access Point auch Mittler in ein weiteres Netz, das sowohl ein Funknetz als auch ein klassisches Kabelnetz sein kann.
- Im Ad-hoc-Modus ist keine Station besonders ausgezeichnet, sondern alle sind gleichwertig. In solchen Netzen ist zwar ein Datenaustausch einfach möglich, jedoch ist kein gezieltes Routing in externe Netze möglich. Dafür lassen sich Ad-hoc-Netze schnell und ohne großen Aufwand aufbauen. Infrastrukturnetze erfordern, implementiert man sie sinnvoll, mehr Planung.

Abbildung 2-9 zeigt den Aufbau eines WLAN im Infrastruktur-Modus, um die Anbindung verschiedener Endgeräte im häuslichen Umfeld an das Internet zu ermöglichen.

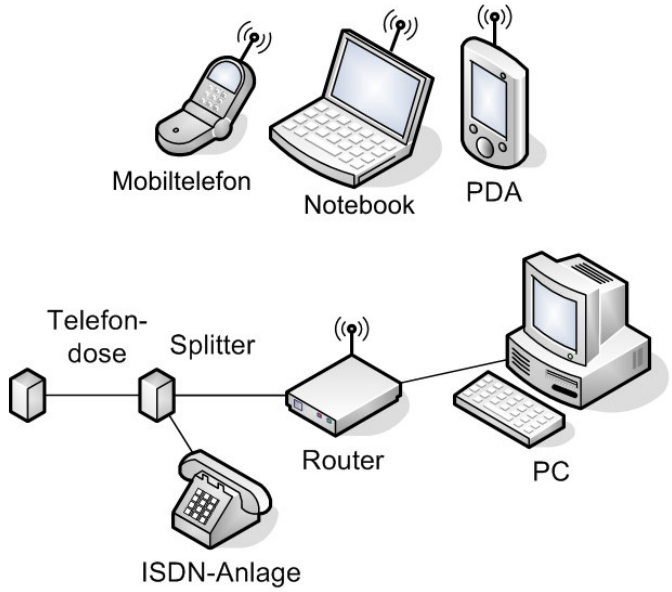

Abbildung 2-9: WLAN im häuslichen Umfeld [WIK05]

Diese Weiterentwicklung der Netztechnologie führt zu einer weiteren Verwischung der Grenzen zwischen häuslicher und mobiler Telearbeit. Die Software zur Unterstützung von Telearbeit muss dieser Entwicklung Rechnung tragen und flexible Zugangsmöglichkeiten auch für unterschiedliche Endgeräte zur Verfügung stellen.

2.4.3 Software

Die Auswahl der Software sollte den Bedürfnissen des damit arbeitenden Menschen bzw. der zu erledigenden Arbeit angepasst sein. In vielen Fällen lassen sich die eingesetzten Programme nicht intuitiv bedienen, was schnell zu Frust beim Anwender führen kann bzw. viele Funktionen ungenutzt lässt. Auch sehr zeitintensive Schulungen und umfangreiche Handbücher können diesen Mangel oft nicht lösen. Untersuchungen ergaben, dass 95 % aller Aufgaben mit nur 5 % der Programmfunktionalitäten erledigt werden [KOW01].

Deshalb ist es sinnvoller, eine gute individuelle Bedienbarkeit zu gewährleisten anstatt viele Funktionalitäten anzubieten, die am Ende ungenutzt bleiben.

Folgende Kategorien von Software werden am Telearbeitsplatz benötigt:

- Systemsoftware
  - Betriebssystem,
  - Netzwerkverbindungen,
  - Sicherheitsmechanismen.
- Anwendungssoftware
  - Office-Anwendungen (Textverarbeitung, Tabellenkalkulation, Grafikbearbeitung, Datenbanken),
  - spezielle Anwendungen (CAD, Programmierumgebungen, Designwerkzeuge).
- Kommunikations- und Kollaborationssoftware
  - synchrone Kommunikation (Videokonferenz, Chat, VoIP),
  - asynchrone Kommunikation (E-Mail, Instant Messaging, News),
  - Shared Workspaces, Dokumentenmanagement, Wissensmanagement,
  - Content- und Application-Sharing, Whiteboard,
  - Kalender, Projektplanung, Meeting-Unterstützung.

Eine weitere Forderung an die Unterstützung von Telearbeit ist, dass die am Büroarbeitsplatz genutzten Programme im gleichen Umfang am Arbeitsplatz zu Hause zur Verfügung stehen sollten. Somit braucht sich der Mitarbeiter nicht umzustellen und kann mit der gleichen Effektivität arbeiten, ob zu Hause oder im Büro. Außerdem werden technische Inkompatibilitäten vermieden. Zentrale Anpassungen der Systemlösungen sollten nicht nur für Telearbeiter, sondern für alle Mitarbeiter in vergleichbaren Funktionen erfolgen.

2.5 Chancen und Risiken der Telearbeit

Wie bei jeder Veränderung des sozialen Gefüges durch innovative Technologien ist auch bei Telearbeit die Betrachtung sowohl positiver als auch nachteiliger Gesichtspunkte notwendig, um eine Bewertung zu erreichen, die über die Begeisterung für neue Arbeitsmöglichkeiten hinaus geht. Deshalb sollen zuerst die Erwartungen der beteiligten Nutzergruppen analysiert und dann Vor- und Nachteile von Telearbeit betrachtet werden.

2.5.1 Erwartungen der einzelnen Nutzergruppen

Die Erwartungen, mit denen Mitarbeiter und Unternehmen an die Einführung von Telearbeit gehen, sind zwangsläufig sehr verschieden. Das sich die verschiedenen Motive für die Umstellung herkömmlicher Arbeitsstrukturen aber zu beiderseitigem Nutzen vereinbaren lassen, wird in den folgenden Abschnitten aufgezeigt.

2.5.1.1 Telearbeiter

Das Bild des Telearbeiters in den Medien wird oft von Müttern mit Kleinkindern geprägt, die durch Telearbeit eine bessere Vereinbarkeit von Beruf und Familie erreichen[24]. Dass es

[24] Übrigens verteilt sich Telearbeit fast gleichmäßig auf die beiden Geschlechter, der Anteil von Männern überwiegt sogar (Vgl. http://www.studie-deutschland-online.de/).

aber noch viele andere Gründe für die Ausübung von Telearbeit gibt, die von den Telearbeitern höher gewichtet werden, zeigt die folgende Aufstellung (Abbildung 2-10).

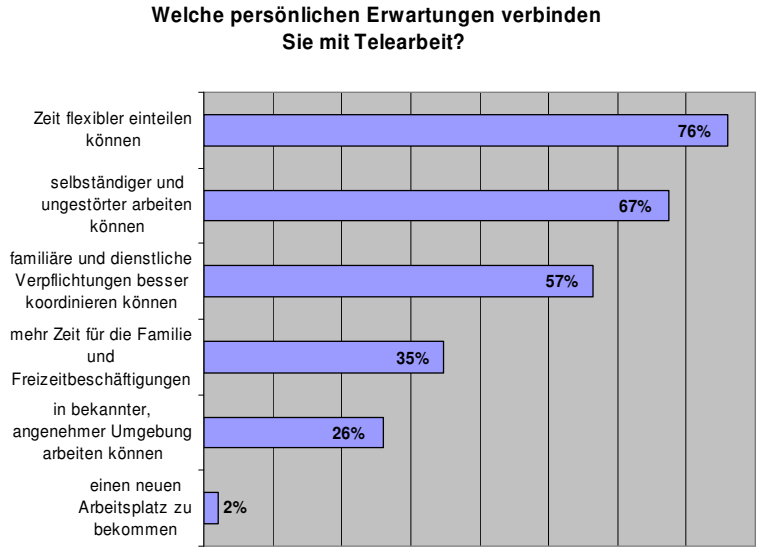

Abbildung 2-10: Erwartungen an Telearbeit[25]

Wie in der Abbildung ersichtlich ist, überwiegen die Erwartungen, sich die Zeit flexibler einteilen und zu Hause ungestörter arbeiten zu können. Aber auch die Vereinbarkeit von Beruf und Familie wird genannt. Weitere positive Erwartungen sind:

- mehr Ruhe und Konzentration am Telearbeitsplatz,
- Fahrtkostenersparnis,
- Zeitersparnis durch Reduzierung der Pendelzeiten,
- erleichterter Wiedereinstieg nach privat bedingten Ausfallzeiten, z.B. Elternzeit.

Die Einführung moderner Arbeitsformen wird meist aber auch von einer Reihe von Befürchtungen begleitet. Viele Telearbeiter befürchten die fehlende Einbeziehung in innerbetriebliche Entscheidungen und das soziale Leben im Unternehmen und damit auch sinkende Karrierechancen. Ein weiteres Problem ist die Anerkennung der Arbeitsleistung durch Vorgesetzte und Kollegen. Oft wird in deutschen Unternehmen noch immer die Anwesenheitszeit als Kriterium von Arbeitsleistung und Leistungsbereitschaft gesehen. Hier muss vor allem im mittleren Management ein Umdenken angestoßen werden, hin zu ergebnisorientierten Führungsstilen wie „*Management by Objectives*“. Es können auch Befürchtungen bestehen, dass Telearbeiter von beruflicher Weiterbildung und Qualifikation ausgeschlossen werden.

Als Gefahr wird oft genannt, dass kein ausreichender Schutz der Privatsphäre und Wohnung des Telearbeiters mehr möglich ist, da der Arbeitgeber ein Weisungsrecht für den Arbeitnehmer hat und auch für die Einhaltung des Arbeitschutzes am häuslichen Arbeitsplatz verantwortlich ist. Diese Gefahr kann aber durch eindeutige Regelungen in Betriebs- oder Individualvereinbarungen ausgeschlossen werden. Manche Arbeitnehmer befürchten auch, dass ihnen ein zusätzlicher Aufwand (auch finanziell) durch Telearbeit entsteht. Dies kann in Einzelfällen eintreten, wenn der Arbeitgeber z.B. keine Erstattung von Telekommunikations-, Strom- und Heizkosten übernimmt, wird aber durch viele nicht-

[25] Befragung von 46 Telearbeitern im Projekt intermobil 2003.

monetäre Vorteile wieder aufgewogen. Weitere positive wie negative Erwartungen von Telearbeitern können in [KOW01] detailliert nachgelesen werden.

2.5.1.2 Arbeitgeber, Auftraggeber

Welche Motive und Erwartungen verbinden aber Unternehmen mit der Einführung von Telearbeit? Auch hier gibt es sehr unterschiedliche Gründe für die Umstellung; als besonders wichtig wird aber meist die Steigerung der Mitarbeitermotivation und der Produktivität eingestuft. Abbildung 2-11 wurde [JäR01] entnommen und zeigt die Wichtung der Erwartungen von Führungskräften.

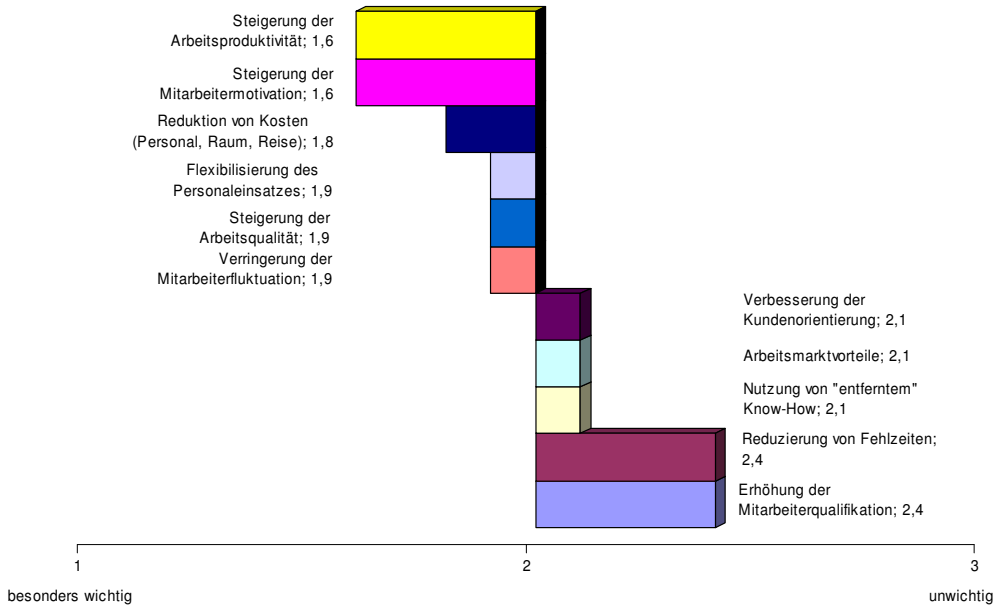

Abbildung 2-11: Erwartungen an die Einführung von Telearbeit aus Sicht der Führungskräfte [JäR01]

An vorderster Stelle stehen die Hoffnungen, dass mit Telearbeit eine Steigerung der Mitarbeitermotivation und damit auch ihrer Arbeitsproduktivität erreicht werden kann. Als eher sekundär werden die Reduzierung von Fehlzeiten und die Erhöhung der Mitarbeiterqualifikation eingestuft. Aber auch Führungskräfte haben nicht nur positive Erwartungen an Telearbeit, sondern auch eine Reihe von offenen Fragen und Befürchtungen, welche da wären:

- Gewährleistung vertrags- und termingerechter Leistungserbringung,
- Überwachung und Kontrolle der Arbeitszeit,
- Störung des innerbetrieblichen Informationsflusses,
- Verfügbarkeit des Arbeitsnehmers bei dringenden Aufgaben,
- Schutz sensibler Daten nach außen, Vertraulichkeit der Datenübermittlung,
- zusätzliche Kosten durch Telearbeit.

Einige der Befürchtungen sind unbegründet, von anderen müssen sich die Verantwortlichen verabschieden und die Chancen zur Umgestaltung starrer konventioneller Arbeitsprozesse zu flexiblen ergebnisorientierten Prozessen nutzen. Andere können wiederum durch geeignete Gegenmaßnahmen (z.B. Einsatz von Verschlüsselung bei der Datenübertragung) von vornherein ausgeschlossen werden.

2.5.1.3 Gewerkschaften, Regierung und allgemeine Interessenvertreter

Nachdem die Gewerkschaften in den Anfangsjahren der Telearbeit noch vor den Gefahren von Telearbeit gewarnt und sogar ein Verbot von Telearbeit gefordert haben, haben sie sich seit Ende der 90er Jahre aktiv am Gestaltungsprozess von Telearbeit und deren Rahmenbedingungen und vor allem an der Ausarbeitung von Telearbeitsvereinbarungen beteiligt. Im Interesse der Arbeitnehmer fordern sie jedoch die Einhaltung folgender Rahmenbedingungen, um die negativen Folgen von Telearbeit so gering wie möglich zu halten.

Geforderte Rahmenbedingungen:

- Mitspracherecht bei der Einführung von Telearbeit (Betriebsverfassungsgesetz),
- Unantastbarkeit der Privatsphäre der Wohnung, Einschränkung der Überwachungs- und Kontrollmöglichkeiten durch den Arbeitgeber,
- keine soziale Isolierung des Telearbeiters, Erhaltung der sozialen und gesellschaftlichen Bindungen im Unternehmen,
- keine "Scheinselbständigkeit" des Telearbeiters.

Wichtig ist, dass sich alle Beteiligten frühzeitig über ihre Intensionen und Erwartungen abstimmen, um so für beide Seiten akzeptable Lösungen zu finden und einige Probleme von vornherein auszuschließen.

2.5.2 Vorteile von Telearbeit

Wegen der Vielzahl verschiedener Telearbeitsformen sind allgemeine Aussagen über Vor- und Nachteile schwierig. Telearbeit bringt derzeit vor allem Vorteile bei qualitativen Faktoren wie Mitarbeiter- und Kundenzufriedenheit. Unstrittiger Vorteil ist z.B. die höhere Zeitsouveränität der Telearbeiter. Hohe Einsparpotenziale lassen sich realisieren, wenn die Organisationsstruktur entsprechend angepasst wird, die Geschäftsprozesse digitalisiert werden und die Telekommunikationskosten durch den Wettbewerb weiter sinken. In einigen Anwendungsbereichen (z.B. Außendienst) sind bereits heute deutliche Einsparungen möglich.

Abbildung 2-12 zeigt die Vorteile von Telearbeit und wie sie von 272 zufällig ausgewählten Unternehmen in einer Befragung der TA Telearbeit [BMB96] gewichtet wurden.

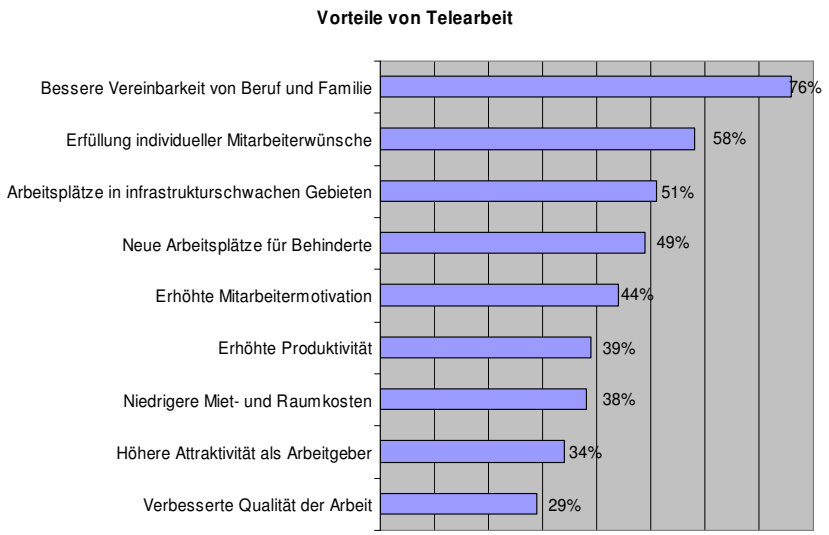

Abbildung 2-12: Vorteile von Telearbeit [BMB96]

Als wichtigster Vorteil wird die bessere Vereinbarkeit von Beruf und Familien eingestuft. Aber auch die Schaffung von Arbeitsplätzen in strukturschwachen Gebieten und für Behinderte wurde als großer Nutzen der Telearbeit gewertet. Weitere Untersuchungen in vorangegangenen Projekten [BHS98a] machten deutlich, dass die Haupteinsparungen durch Telearbeit im Bereich Büromietkosten und Fahrtkosten sowie Fahrtzeit zwischen Wohnung und Büro erzielt werden können. Letztere Kostenreduzierungen kommen hauptsächlich dem Arbeitnehmer zugute, können aber über einen Interessenausgleich zwischen Arbeitgeber und Arbeitnehmer auch Vorteile für das Unternehmen bringen. Die bei alternierender Telearbeit zusätzlich entstehenden Kosten für doppelte Arbeitsplätze können durch so genanntes „Desk Sharing“, die alternierende Nutzung des Büroarbeitsplatzes durch zwei oder mehrere Mitarbeiter, gesenkt werden.

Ansonsten lassen sich in konventionellen Arbeitsprozessen für beide Seiten nur selten große Einsparungen erzielen. Dies gilt ganz besonders für Pilotversuche, an denen nur vergleichsweise wenige Mitarbeiter teilnehmen und optimale Organisationsformen und Arbeitsabläufe erst gefunden werden müssen. In dieser Phase behalten die Telearbeiter in der Regel ihren persönlichen betrieblichen Arbeitsplatz; die Kosten für den Telearbeitsplatz und die Telekommunikationskosten fallen zusätzlich an. Einsparungen für das Unternehmen ergeben sich aus der höheren Produktivität und dem nachweislich geringeren Krankenstand bei Telearbeitern. Laut dem Marktforschungsinstitut Gallup liegt die Produktivität bei Telearbeitern um 22 bis 45 Prozent höher - "infolge von weniger Unterbrechungen, geringeren Ausfallzeiten wegen Witterungsbedingungen oder kranken Kindern und dem Wegfallen des Pendelverkehrs zum Arbeitsplatz und zurück." [Wit03].

2.5.3 Nachteile von Telearbeit

In der Diskussion zur Telearbeit werden eine ganze Reihe Problemfelder genannt. Abbildung 2-13 zeigt die Gewichtung der Nachteile, wie sie bei der in Kapitel 2.5.2 bereits erwähnten Befragung der TA Telearbeit festgestellt wurden [BMB96].

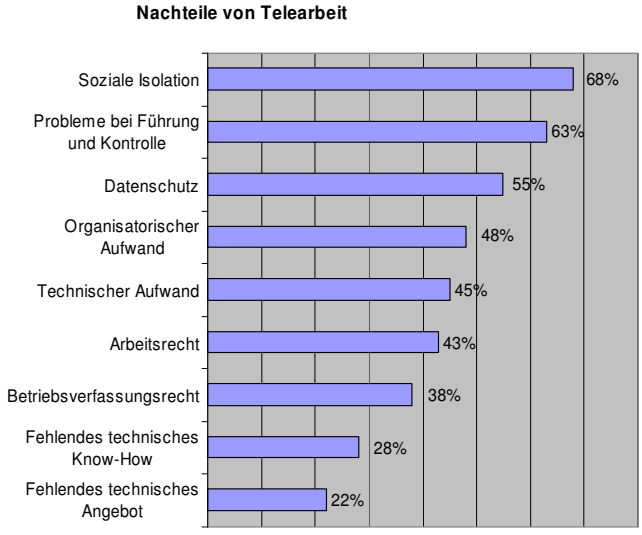

Abbildung 2-13: Nachteile von Telearbeit [BMB96]

Dabei wird immer wieder betont, dass die Lösung der nichttechnischen Probleme vor allem "im Kopf" stattfinden muss und dass eine wesentliche Hemmschwelle in der Bereitschaft des mittleren Managements liegt, die direkte (Anwesenheits-) Kontrolle der Mitarbeiter durch Vorgabe von Zielen zu ersetzen. Telearbeit ist der Einstieg in flachere Hierarchien und in die Führung durch Zielsetzung und Eigenmotivation. Diese notwendigen tie-

fen Veränderungen in Management- und Arbeitstraditionen sind das eigentliche Problem der Durchsetzung von Telearbeit.

2.5.4 Fazit

Insgesamt zeigen die Erfahrungen, dass sich die starken sozialen Ängste nicht bestätigen und dass es genügend Möglichkeiten gibt, den realen Gefahren entgegenzutreten. Wichtig scheint, dass die Unternehmenskultur dahingehend geändert wird, dass die Kommunikation über Online-Dienste eine größere Bedeutung im gesamten Unternehmen erlangt und somit Telearbeiter nicht vom sozialen Leben des Unternehmens ausgeschlossen werden. Dazu gehört es aber auch, dass genügend Gelegenheiten geschaffen werden, bei denen direkter sozialer Kontakt mit den Kollegen gegeben ist, unter Umständen auch außerhalb der Arbeit.

Telearbeit bietet nicht nur Chancen, Unternehmens- und Verwaltungsprozesse effektiver und wirtschaftlicher zu gestalten, sondern eröffnet auch enorme Potenziale hinsichtlich einer markt- und kundenorientierten Organisationsentwicklung. Der Einsatz von Telearbeit erhöht die Flexibilität und damit die Wettbewerbsfähigkeit von Unternehmen.

2.6 Anforderungen an eine universelle Arbeitsumgebung

2.6.1 Funktionale Anforderungen

Die wichtigste Anforderung an eine für Telearbeit geeignete Tätigkeit ist die physikalische Auslagerungsfähigkeit der Arbeitsmittel. Das heißt, die notwendigen Werkzeuge und Dokumente müssen durch einen Online-Zugriff verfügbar gemacht werden können. Dieser Zugriff sollte dabei keine speziellen Kenntnisse und spezielle Software erfordern, sondern am besten durch einheitliche und leicht handhabbare webbasierte Dienste erfolgen und dennoch alle erforderlichen Sicherheitskriterien erfüllen. Besonders wichtig ist bei Telearbeit auch die Abbildung der in einer herkömmlichen Büroumgebung vorhandenen Dienste wie Hauspost, Rundschreiben, Technischer Support oder Dienstbesprechungen auf ortsunabhängige elektronische Dienste (siehe Abbildung 2-14).

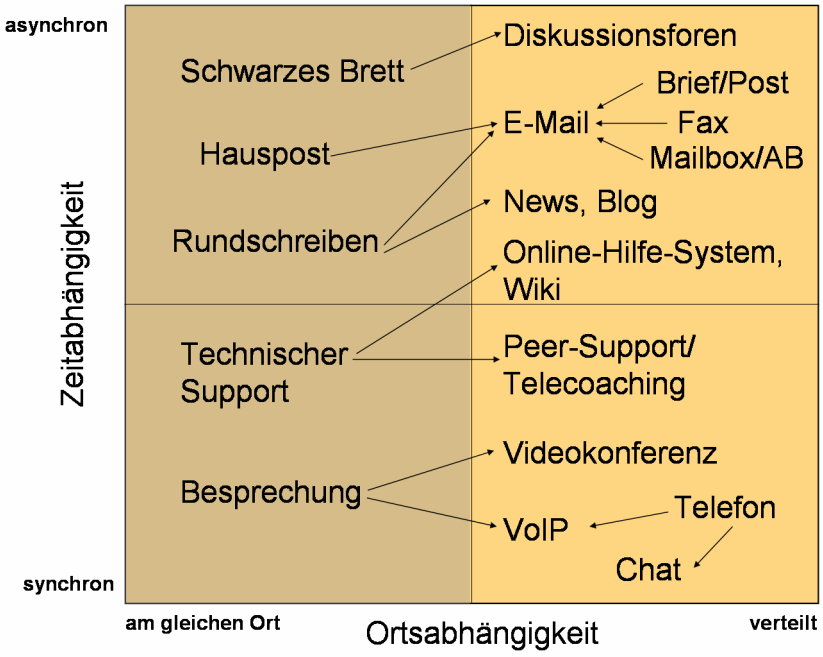

Abbildung 2-14: Kommunikations- und Kooperationswerkzeuge bei Telearbeit

Im Folgenden sollen die wichtigsten funktionalen Anforderungen an eine universelle Telearbeitsumgebung aufgeführt und erörtert werden.

2.6.1.1 Universelle Einsetzbarkeit

Die Arbeitsumgebung soll in der Hinsicht universell einsetzbar sein, dass sie für alle Telearbeitsformen und auch im „normalen“ Büroalltag gleichermaßen nutzbar ist. Der Aufbau der Arbeitsumgebung soll so flexibel sein, dass man sie an alle erdenklichen Einsatzszenarien anpassen kann. Außerdem soll der Zugriff auf sie an keine Betriebssystemplattform oder an bestimmte Endgeräte gebunden sein.

2.6.1.2 Modularer Aufbau

Die Architektur soll es erlauben, verschiedenste Anwendungen als unabhängige Komponenten nach dem Baukastenprinzip zu integrieren, unabhängig von der Programmiersprache, in der sie realisiert sind, und unabhängig von der Systemplattform, auf der sie sich befinden. Die intelligente Verknüpfung geeigneter standardisierter Software-Komponenten ist der zukünftige Software-Ansatz. Die Entwicklung geht weg vom Spaghetticode und Monolithen hin zur Komponenten- und serviceorientierten Architektur.

2.6.1.3 Webbasierter Zugang

In den letzten Jahren ist ein deutlicher Trend in der Entwicklung von Telearbeitsanwendungen von so genannten „*Fat Clients*“[26] hin zu „*Thin Clients*“[27] zu beobachten. Beim Fat-Client-Ansatz befinden sich die Daten und Anwendungsprogramme beim Client, also am Telearbeitsplatz. Eine Serververbindung wird nur zum Synchronisieren der Daten mit dem Unternehmen oder Büroarbeitsplatz benötigt. Das Ziel dieses Ansatzes war es, so wenig Netzzeit wie möglich zu verbrauchen. Durch die stetig sinkenden Preise für Online-Gebühren hat sich in den letzten Jahren ein Paradigmen-Wechsel zur "*always on*"-Philosophie unter Nutzung permanenter Netzzugänge, so genannter *Flat-Rates*, vollzogen. Der Thin-Client-Ansatz macht sich dies zu Nutze, indem die Daten und Programme auf dem Server verbleiben und direkt dort verarbeitet werden. Dadurch werden auf Client-Seite keine (oder nur sehr wenige) Programme und Daten benötigt.

Browserbasierte Thin-Client-Anwendungen können problemlos verwaltet sowie auf unterschiedlichsten Desktops ausgeführt werden und haben keinerlei Auswirkungen auf den Zustand des Clientcomputers. Trotz dieser Vorzüge ist das browserbasierte Modell jedoch alles andere als perfekt. Spezielle Clientanwendungen bieten eine vielseitigere Benutzeroberfläche sowie den Zugriff auf die lokalen Datenträger und die lokalen APIs[28], wodurch dem Entwickler zusätzliche Möglichkeiten zur Verfügung stehen und den Benutzern eine größere Funktionsvielfalt geboten werden kann. Da sie lokal auf dem Clientcomputer ausgeführt werden, können Fat-Client-Anwendungen zudem die verfügbaren Ressourcen besser nutzen, das Problem von Netzwerkwartezeiten vermeiden und dem Benutzer das Arbeiten im Offline-Modus ermöglichen.

---

[26] „Fat Client“ wird in der Literatur oft auch als „Rich Client“ bezeichnet. Bei diesem Ansatz wird die eigentliche Verarbeitung der Daten vor Ort auf dem Client vollzogen und der Client stellt die graphische Benutzeroberfläche (GUI) selbst zur Verfügung.

[27] „Thin Client“ bezeichnet ein Anwendungsprogramm, das seine Daten möglichst vollständig von einem Server bezieht. Das schließt auch die graphische Benutzeroberfläche mit ein. Thin Clients werden meist auf Basis von Browsern realisiert.

[28] APIs (Application Programming Interfaces) - Schnittstellen für die Anwendungsprogrammierung.

Fat-Client-basierte Anwendungen erfordern aber die Installation spezieller Softwarekomponenten zur Verwirklichung der entsprechenden Funktionalität und stehen mit anderen Netzknoten über spezielle Ports in Verbindung. Damit ist die Integration in bestehende Sicherheitskonzepte nicht ganz unproblematisch. Es müssen Firewalls rekonfiguriert bzw. viele zusätzliche Ports geöffnet werden. Dadurch steigt die Zahl der Eintritts- und Angriffspunkte des Systems. Weiterer Nachteil des Fat-Client-Ansatzes ist der große Aufwand bei der Wartung der Anwendungsprogramme, vor allem wenn man sich vorstellt, dass ein Unternehmen eine Vielzahl von Telearbeitern beschäftigt. Beim Thin-Client-Ansatz werden die Anwendungen serverseitig gewartet und installiert und stehen dann gleichzeitig einer Vielzahl von Beschäftigten zur Verfügung.

Bei einigen Telearbeitsformen wie der On-Site-Telearbeit wechselt der Arbeitsplatz des Nutzers sogar öfter und unternehmensfremde Rechentechnik wird genutzt. Dann ist es unablässig, dass die Anwendungen ohne großen Installationsaufwand überall benutzt werden können. Das spricht für die Umsetzung des Thin-Client-Ansatzes und damit für die Nutzung webbasierter Anwendungen.

2.6.1.4 Prozessorientierung

Wichtiges Kriterium für eine erfolgreiche Telearbeit ist die Einbindung des Telearbeiters in die unternehmensinternen Abläufe. Deshalb ist die prozessorientierte Abbildung der Workflows im Unternehmen durch die Arbeitsumgebung eine wichtige Voraussetzung für eine effektive Unterstützung des Telearbeiters.

Das entfernte Bearbeiten einer Aufgabe muss dahingehend unterstützt werden, dass alle zur Bewältigung einer Aufgabe erforderlichen Ressourcen, wie bspw. Daten oder Werkzeuge, dem beteiligten Telearbeiter am jeweiligen Arbeitsort zur Verfügung stehen. Dafür sind Dienste zum Workflow-, Ressourcen- und Dokumentenmanagement notwendig. Diese Dienste müssen entsprechend der abzubildenden Workflows zusammengestellt und integriert werden. Dabei sollen dem Telearbeiter Hilfestellungen durch die Bereitstellung vordefinierter Entwurfsmuster gegeben werden.

2.6.1.5 Kommunikations- und Kooperationsförderlichkeit

Zu den wichtigsten Ansprüchen der Telearbeiter gehören vor allem eine schnelle Kommunikation mit Kollegen und Kunden und eine gute technische Betreuung. Aus technischer Sicht muss daher eine geeignete Unterstützung der entfernten Arbeit und der Gruppenarbeit realisiert werden, die insbesondere die Aspekte Kommunikation, Kollaboration und Koordination abdeckt.

Die *Kommunikation* zwischen Auftraggeber und Auftragnehmer und zwischen Personen, die gemeinsam eine Aufgabe bewältigen, ist eine elementare Voraussetzung für erfolgreiche Telearbeit und muss vor allem dann gewährleistet werden, wenn sich die Personen an verschiedenen Orten befinden. Prinzipiell kann zwischen synchroner und asynchroner Kommunikation unterschieden werden. Synchrone Kommunikation ist die natürliche Form der Interaktion zwischen Menschen und wird beispielsweise beim Telefonieren oder bei einer Videokonferenz durchgeführt. Asynchrone Kommunikation beinhaltet zeitliche Verzögerungen und liegt bei den meisten Formen des elektronischen Nachrichtenaustausches wie z. B. E-Mail vor. Während asynchrone Kommunikation für einfache Formen der Telekooperation ausreichend ist, sind komplexe und anspruchsvolle Aufgaben nur unter Einsatz synchroner Kommunikationswerkzeuge zu realisieren. Eine Computerunterstützung für Telearbeitsszenarien sollte daher beide Kommunikationsformen unterstützen.

Beim gemeinsamen Bearbeiten einer Aufgabe, auch mit dem Begriff *Kollaboration* bezeichnet, muss sichergestellt werden, dass alle beteiligten Telearbeiter die Daten und Werkzeuge am Arbeitsplatz zur Bearbeitungszeit zur Verfügung haben. Für die Sicherung der Konsistenz der bearbeiteten Daten und Dokumente bei der Bearbeitung durch mehrere Telearbeiter ist das Dokumentenmanagement verantwortlich.

Um eine effiziente Erfüllung der Aufgaben zu gewährleisten, muss eine *Koordination* der Aktivitäten realisiert werden. Kooperatives Arbeiten mehrerer Personen ist potenziell konfliktbehaftet, da beispielsweise beim parallelen Bearbeiten eines Dokumentes widersprüchliche Änderungen auftreten können. Bei der Zusammenarbeit in größeren Gruppen (ab vier Personen) wird sogar von einer "Produktionsblockade" gesprochen, da die Beseitigung der Inkonsistenzen soviel Aufwand erfordert, dass nahezu keine produktive Arbeitsleistung mehr möglich ist. Die Koordination der einzelnen Aktivitäten jedes Kooperationspartners einschließlich der Synchronisation des Zugriffs auf gemeinsame Ressourcen ist in gewissem Sinne zwar kein Bestandteil der Telearbeit, aber für ihre Durchführung notwendig. Zur Bewältigung dieser Aufgabe müssen Aufgaben- und Dokumentenmanagement eng zusammenarbeiten.

Eine universelle Telearbeitsumgebung sollte nicht nur die unternehmensweite Kommunikation unterstützen, sondern die Telearbeiter auch bei der Entscheidungsfindung mit einbeziehen und die Koordination täglich anfallender Aufgaben, Termine sowie Dokumente ermöglichen. Zielsetzung ist dabei, eine gemeinsame Datenbasis bereitzustellen, auf die alle Nutzer, auch die Büroarbeiter, ortsunabhängig zugreifen können. Die benötigten Informationen sollten allen Beteiligten sofort zur Verfügung stehen und die Betrachtung der Daten aus verschiedenen Sichtweisen und die Bearbeitung entsprechend der ihnen zugeteilten Rechte ermöglichen.

2.6.1.6 Standardisierung des Datenaustauschformats

Damit ein Datenaustausch zwischen den verschiedenen Anwendungen möglich ist, muss ein standardisiertes Datenaustauschformat genutzt werden. Die *Extensible Markup Language* oder kurz XML, die ursprünglich als reine Markup-Sprache gedacht war, hat sich mittlerweile als universelle Datenaustauschsprache durchgesetzt.

2.6.1.7 Individualisierbarkeit und Personalisierbarkeit

Die Arbeitsumgebung soll an die Bedürfnisse und Fähigkeiten des einzelnen Nutzers individuell angepasst werden können. Dies schließt die individuelle Festlegung von Oberflächeneigenschaften wie Farbgestaltung und Anordnung von Bedienelementen ebenso mit ein wie die flexible Integration von Diensten und Anwendungen.

2.6.1.8 Adaptierbarkeit

Zukünftig wird die Bereitstellung von Diensten für mobile Endgeräte immer wichtiger. Durch die stete Weiterentwicklung mobiler Hardware ergeben sich neue interessante Einsatzszenarien. Mobile Telearbeiter können so effektiver in die Geschäftsprozesse eingebunden und mit den notwendigen Informationen versorgt werden. Dies erfordert die Möglichkeit der flexiblen Anpassung der Benutzeroberfläche und der angebotenen Funktionalität an verschiedene Endgeräte und deren spezifische Eigenschaften.

2.6.2 Nichtfunktionale Anforderungen

Die geplante universelle Lösung stellt ein hochkomplexes Softwareprodukt dar, das unabhängig von seiner Funktionalität verschiedensten Anforderungen aus Sicht des Software-Engineering genügen muss. Diese so genannten nichtfunktionalen Anforderungen werden in den folgenden Abschnitten detailliert beschrieben.

2.6.2.1 Offenheit der Systemimplementierung

Der Begriff der Offenheit wird je nach Kontext unterschiedlich gebraucht, so dass es in der Praxis oft zu einer Verwirrung kommt. Deshalb wird an dieser Stelle zunächst eine kurze Klärung des Begriffs vorgenommen. Der Begriff „Offenes System“ wurde von der IEEE[29] im Rahmen des POSIX-Projektes [POS92] definiert als „ein System, das in ausreichendem Maße offen gelegte Spezifikationen für Schnittstellen und dazugehörige Formate implementiert, damit entsprechend gestaltete Anwendungssoftware

- auf eine Vielzahl verschiedener Systeme (mit Anpassungen) portiert werden kann (Portabilität),
- mit anderen heterogenen Anwendungen lokal und über heterogene Netzwerke entfernt zusammenarbeiten kann (Interoperabilität),
- mit Benutzern in einer Art interagiert, die das Wechseln der Benutzer zwischen verschiedenen Systemen erleichtert (Einhaltung von Standards der Dialoggestaltung).“

Grundlegende Voraussetzung für Interoperabilität und Portabilität ist die Nutzung offener Standards bei der Implementierung der Anwendung.

2.6.2.2 Plattformunabhängigkeit

Da der Arbeitsplatz des Nutzers bei On-Site-Telearbeit regelmäßig wechselt, ist es unablässig, dass die Systeme ohne großen Installationsaufwand überall benutzt werden können. Arbeitsoberfläche und Funktionsumfang sollten an jedem Rechner identisch sein. Um möglichst flexibel bei der Wahl des Arbeitsrechners zu sein, sollte die Applikation plattformunabhängig ausführbar sein.

2.6.2.3 Verteilbarkeit des Systems

Durch die Verteilung des logischen Systems auf mehrere physikalische Knoten soll eine Erhöhung der Performance und Verfügbarkeit ermöglicht werden. Die verschiedenen Dienste sollen lose an die Arbeitsumgebung gekoppelt werden und dadurch flexibel durch andere gleichwertige Dienste ersetzt werden können.

2.6.2.4 Transparenz

Für den Nutzer soll es weitestgehend transparent sein, von wem der genutzte Dienst erbracht wird und wo sich der Diensterbringer im Netz befindet. Weiterhin sollen komplexe Dienste nach außen transparent über eine Schnittstelle erreichbar sein, so dass die eigentliche Komplexität dem Nutzer verborgen bleibt.

---

[29] Das IEEE (Institute of Electrical and Electronics Engineers) ist ein weltweiter Verband von Ingenieuren aus den Bereichen Elektrotechnik und Informatik. Es bildet Gremien für die Standardisierung von Technologien, Hardware und Software.

2.6.2.5 Skalier- und Erweiterbarkeit

Die Arbeitsumgebung soll möglichst flexibel erweiterbar und das Gesamtsystem an veränderte Anforderungen, z.B. die Erhöhung der Nutzeranzahl, anpassbar sein. Die Erweiterbarkeit eines Systems hängt maßgeblich davon ab, ob die Fundamente des Systems weit akzeptierte Standards sind und es auch Produkte und Werkzeuge gibt, die diese Standards unterstützen.

2.6.2.6 Wiederverwendbarkeit

Gefordert sind Teilsysteme oder Komponenten, die in anderen Umgebungen weiterhin arbeitsfähig sind und in verschiedenste Infrastrukturen integriert werden können. Der Nutzen für den Entwickler besteht darüber hinaus in der Wahrung der geleisteten Investitionen sowie der Beibehaltung vertrauter Systeme und Umgebungen.

2.6.2.7 Verfügbarkeit

Das System muss eine gewisse Ausfallsicherheit gewährleisten. Das heißt, falls z.B. ein Dienst ausfällt, darf das Gesamtsystem nicht davon beeinträchtigt werden, sondern muss in einem definierten Zustand weiterarbeiten. Die Verfügbarkeit einzelner Dienste ist durch den jeweiligen Service Provider zu garantieren.

2.6.2.8 Zuverlässigkeit

Mit der zunehmenden Anzahl von Telearbeitsplätzen wächst auch die Komplexität des Systems und damit die Fehleranfälligkeit. Gleichzeitig steigt bei verteilter Arbeit die Abhängigkeit von der Technik. Fehler können nicht mehr so einfach durch direkte menschliche Kommunikation überbrückt werden. Technische Zuverlässigkeit und ausgebaute Servicestrukturen sind deshalb geboten. Verteilte Systeme sind deutlich fehleranfälliger als ein zentrales System. Die Software-Lösung muss auf den Ausfall einzelner Hardware-Komponenten vorbereitet sein und eine Ausweichlösung anbieten. Handelt es sich um ein verteiltes System, muss das System auf eine Übergabe von IP-Adressen und des Transaktionskontextes, den so genannten *Failover*, vorbereitet sein (Ausfallsicherheit). Bei einem Komplettausfall des Systems (z.B. des Portalservers) muss die Möglichkeit bestehen, das System in den gleichen Zustand zu versetzen, in dem es sich vor dem *Breakdown* befunden hat. Kontextinformationen und Anwendungsdaten müssen so gut wie möglich rekonstruierbar sein.

Neben den allgemeinen Anforderungen wie Interoperabilität, Zuverlässigkeit und Skalierbarkeit muss eine universelle Arbeitsumgebung für Telearbeiter auch noch weitere nichtfunktionale Anforderungen erfüllen.

2.6.2.9 Benutzbarkeit

Mensch-Maschine-Schnittstellen werden zunehmend zum Problem, wenn Telearbeit auf nicht speziell geschulte, breite Nutzerschichten trifft und gleichzeitig die Vielfalt der technischen Möglichkeiten steigt. Einfachheit und Intelligenz der Systeme wird damit zur Grundvoraussetzung, um sich als Telearbeiter in der Komplexität der angebotenen Lösungen zurechtzufinden. Die Technik muss „intuitiv“ auf den Nutzer eingehen, fehlertolerant reagieren und differenzierte Eingabemedien akzeptieren. Gleichzeitig muss sie lernfähig sein und Nutzerprofile erkennen und unterstützen.

2.6.2.10 Wartbarkeit

Die Wartbarkeit von Systemen ist ein wichtiges Kriterium bei der Bewertung von Kostenvorteilen durch den Einsatz einer bestimmten Lösung. Zur Wartbarkeit gehört, dass auch Personal ohne detaillierte Kenntnisse des Innenlebens des Systems mit einfachen Mitteln die Funktionstüchtigkeit überprüfen und Fehler beheben kann. Dazu sollten auf dem Markt entsprechende Werkzeuge zur Verfügung stehen, weshalb das Systems unbedingt auf Standards aufbauen sollte. Außerdem sollte eine zentrale Wartung aller, auch der verteilten Einzelkomponenten möglich sein.

2.6.2.11 Sicherheit

Mit der angestrebten Lösung soll der ortsunabhängige Zugang zu Unternehmensdaten und -anwendungen ermöglicht werden. Da der Zugriff auch über öffentliche Netze erfolgen kann, ist der Gewährleistung von Sicherheit besondere Beachtung zu schenken. Ein unberechtigter Zugriff auf sensible Unternehmensdaten muss dabei unbedingt verhindert werden. So gehören die Verschlüsselung von Nachrichtenströmen und die Authentifizierung der berechtigten Nutzer zu den Grundvoraussetzungen für Unternehmenslösungen. Diese bedürfen differenzierter Zugriffsmechanismen, die allen an einem Arbeitsprozess beteiligten Mitarbeitern den abgestuften Zugriff auf die für sie relevanten Teilmengen der Informationen und Dokumente im Rahmen einer skalierbaren Rechteverwaltung ermöglichen. Basis dafür kann nur ein feingranulares hierarchisches Gruppen- und Rollenmanagement sein.

KAPITEL 3
ANALYSE UND KLASSIFIZIERUNG VORHANDENER LÖSUNGEN ZUR UNTERSTÜTZUNG VON TELEARBEIT

"Der vernünftige Mensch passt sich der Welt an. Der unvernünftige Mensch besteht darauf, dass sich die Welt nach ihm zu richten hat. Deshalb hängt jeder Fortschritt von dem unvernünftigen Menschen ab."

George Bernard Shaw (1856-1950)

In diesem Kapitel werden vorhandene Softwarelösungen auf Ihre Nutzbarkeit im Rahmen von Telearbeit untersucht und klassifiziert. Es existiert eine Fülle von kommerziellen und frei verfügbaren Anwendungen, die unter bestimmten Voraussetzungen für Telearbeit genutzt werden können, z.B. Groupware- oder Workflow-Applikationen.

Diese Anwendungen wurden aus verschiedenen Beweggründen und für unterschiedliche Anwendungsszenarien entwickelt. Grundlegend kann man drei Kategorien von Anwendungen unterscheiden, die zur Unterstützung von Telearbeit genutzt werden können.

- Anwendungen zur Unterstützung entfernten Arbeitens
- Systeme zur Unterstützung kooperativen Arbeitens
- Integrationslösungen

Anwendungen zur Unterstützung entfernten Arbeitens erlauben den Fernzugriff auf Daten und Ressourcen und die Fernsteuerung entfernter Systeme. Systeme zur Unterstützung kooperativen Arbeitens sind vor allem für die Zusammenarbeit in Teams geeignet und ermöglichen die Kommunikation und Kollaboration der beteiligten Gruppenmitglieder. Integrationslösungen dienen der Zusammenführung verschiedenster Anwendungen, teilweise auch über Unternehmensgrenzen hinweg, und verfolgen dabei entweder einen benutzerorientierten oder einen prozessorientierten Integrationsansatz.

Zum Vergleich wurden die in Kapitel 2 festgelegten Anforderungen auf diese Lösungen abgebildet, um eine Abgrenzung der vorliegenden Arbeit gegenüber anderen Ansätzen und Systemen vornehmen zu können.

3.1 Unterstützung entfernten Arbeitens

In diesem Kapitel sollen Lösungen vorgestellt werden, die einen direkten Zugriff auf entfernte Rechner ermöglichen, was vor allem in der Fernwartung von Rechnersystemen eine große Rolle spielt, aber auch für Telearbeit geeignet scheint. Durch den Transfer von Steuereingaben durch Maus und Tastatur in einer Richtung und vom Bildschirminhalt in der anderen Richtung wird die Fernsteuerung eines entfernten Rechners unterstützt.

3.1.1 Terminaldienste

Während früher Terminaldienste vor allem für den Zugang zu großen Server-Systemen konzipiert wurden, hat sich in den letzten Jahren auch der Fernzugriff auf Desktop-Rechner etabliert. Dabei ermöglicht die Terminal-Technik den Fern-Zugang zu Daten und Ressourcen eines Rechners und damit auch meist den Zugang zum Intranet eines Unternehmens. Der Vorteil dieser Lösung ist, dass unternehmensinterne Daten den

geschützten Bereich des lokalen Netzwerkes gar nicht erst verlassen. Hierzu wird ein Terminal-Server installiert, der intern als Proxy für den externen Nutzer fungiert. Die Steueranweisungen werden in Form von Tastatur und Maus-Eingaben des externen Nutzers vom Terminal-Server empfangen und die Bildschirm-Darstellung des Rechners zum externen Nutzer gesendet. Die Übertragung dieser Informationen wird bei Nutzung von Microsoft-Betriebssystemen bereits protokollseitig verschlüsselt, sodass fremde Kenntnisnahme oder Manipulation praktisch ausgeschlossen werden kann. Für die Steuerung dieses Proxies werden auf externen Systemen sog. Terminalclients verwendet, die für die Kommunikation mit dem Terminal-Server sorgen. Diese Clients stehen teilweise als Komponente des Betriebssystems oder als zusätzliche Softwarelösung für die unterschiedlichsten Systeme zur Verfügung.

Der Terminal-Server ist eine in vielen Server- und auch einigen Desktop-Betriebssystemen integrierte Funktion, die es mehreren Benutzern erlaubt, sich gleichzeitig über ein Netzwerk an dem Server anzumelden und dessen Ressourcen zu nutzen. Vom Terminal-Server werden die reinen Bildinformationen zur Darstellung des Desktops an den Client übertragen und so auf dem Telearbeitsrechner in einem Fenster eine vollständige Benutzeroberfläche des entfernten Rechners dargestellt. Umgekehrt werden alle am Telearbeitsplatz ausgelösten Tastatur- und Mauseingaben an den Terminal-Server gesendet. Die Benutzer können so die aus dem Büroalltag bekannten Applikationen *remote* nutzen und im Rahmen der dienstlichen Notwendigkeit auf andere Ressourcen im Netzwerk zugreifen.

Beim Zugriff auf Microsoft-Betriebssysteme wird das *Remote Desktop Protocol* (RDP) verwendet, welches alle Bild- und Tastaturströme verschlüsselt überträgt. Die eigentliche Datenverarbeitung findet im geschützten Serverbereich statt. Somit verlassen die Anwendungsdaten nicht das geschützte Unternehmensnetzwerk. Damit der Terminal-Server nicht als Schlupfloch von externen Angreifern genutzt werden kann, sollten zusätzliche Sicherheitsmechanismen wie *Virtual Private Network* (VPN) eingesetzt werden. Ein weiterer Vorteil des RDP von Microsoft ist die Anbindung lokaler Ressourcen des Client-Rechners wie z.B. Drucker oder Laufwerke an den ferngesteuerten Computer. So kann aus einer *Remote Session* heraus problemlos auf dem häuslichen Printer gedruckt oder auf lokale Verzeichnisse auf dem Home-Rechner zugegriffen werden.

Die meisten modernen Windows-Fassungen beinhalten mittlerweile ein Clientprogramm zur Nutzung des *Remote Desktop Protocol*. Den Client für den Zugriff auf den XP-Desktop stellt Microsoft auch für ältere Windows-Versionen und Mac OS X zur Verfügung, Linux- und Unix-Nutzer können auf das Open-Source-Projekt *rdesktop* zurückgreifen. Mac OS X enthält zwar den so genannten „*Apple Remote Desktop*“, aber das Client-Programm für den Zugriff kann nur als Teil einer teuren kommerziellen Management-Software genutzt werden [EnS05].

Linux- und Unix-Systeme nutzen dagegen das Open-Source-Projekt VNC (*Virtual Network Computer*) als Standardlösung für die Fernsteuerung von entfernten Rechnern. Es deckt derzeit fast die ganze Palette der aktuellen Betriebssysteme, vom Unix-Server über die Windows- und MacOS-Desktops bis zum Java-fähigen Telefon, ab [EnS05], hat aber den Nachteil, dass keinerlei Verschlüsselungsmechanismen eingebaut sind und sich der Nutzer deshalb selbst um die Sicherung der Datenübertragung kümmern muss.

3.1.2 Weitere Remote-Access-Lösungen

Neben diesen betriebssysteminternen oder zumindest -nahen Programmen gibt es mittlerweile eine Reihe von kommerziellen Anwendungen, die unter den Bezeichnungen „*Remote Access*“ oder „*Remote Control*“ verschiedene Funktionen wie Fernsteuerung und Datenübertragung unter einer einheitlichen Oberfläche zusammenfassen. Vertreter wie

PCAnywhere[30] von Symantec, GoToMyPc[31] von Citrix oder Timbuktu[32] von Netopia seien hier nur als bekannteste Beispiele erwähnt. Nachteil dieser Lösungen ist ebenfalls, dass eine Client-Anwendung installiert werden muss. Mittlerweile gibt es auch schon webbasierte Lösungen wie z.B. *PositivePRO WebTop* von Positive Networks[33], die einen Zugriff über die Browser-Schnittstelle auf Windows-Rechner erlauben, wenn auf diesen der *Remote Desktop Agent* installiert ist.

Als nachteilig erweist sich die Beschränkung der Bilddaten auf ein bestimmtes Ausgabeformat, um die zu übertragende Datenmenge zu verringern. Dabei werden vor allem Abstriche bei der Farbdarstellung gemacht. Diese kann bei Tätigkeiten, die eine gute farbliche Ausgabe benötigen, wie z.B. Bildbearbeitung, CAD- und andere Entwurfsarbeiten, zu erheblichen Einschränkungen führen. Solche Tätigkeiten können in der Regel nicht mit Terminaldiensten durchgeführt werden.

Störend ist ebenfalls die verzögerte Reaktion auf Maus- und Tastatureingaben, die einerseits durch die fehlende Bandbreite bei der Übertragung der großen Menge von Bildschirmdaten und andererseits durch die zur Verschlüsselung des Datenverkehrs blockierte Rechenzeit hervorgerufen wird. Ein weiterer Nachteil ist die Systemabhängigkeit der Lösungen und die Notwendigkeit der Installation einer Client-Anwendung. Dies widerspricht der Forderung nach Plattform-Unabhängigkeit und Flexibilität.

3.2 Unterstützung kooperativen Arbeitens

Durch die wachsende Bedeutung teamorientierter Managementkonzepte und die zunehmende Akzeptanz von Computernetzwerken rückte die informationstechnische Unterstützung kooperativen Arbeitens, auch als *Computer Supported Cooperative Work* (CSCW) bezeichnet, etwa Mitte der 80er Jahre in den Blickpunkt der Forschung. Wurden anfänglich isolierte Einzelanwendungen zur Kommunikation und Kooperation in Gruppen entwickelt, fassen heutige Systeme eine Vielfalt von Funktionalitäten in ganzen CSCW-Suites zusammen [BaS00]. Durch die zunehmende Ausweitung von Kooperation auf unternehmensübergreifende Benutzergruppen und die zusätzliche Integration anderer geschäftskritischer Anwendungen stehen heutige Systeme vor neuen Herausforderungen hinsichtlich der Flexibilität und Interoperabilität. Dabei bilden das Internet und die darauf aufbauenden Standards und Protokolle die kommunikationstechnische Grundlage für die Zusammenarbeit von Mitarbeitern, sei es innerbetrieblich oder über Unternehmensgrenzen hinweg. Abbildung 3-1 verdeutlicht die Entwicklung von CSCW-Systemen, die einerseits zu einer deutlichen Erweiterung des Benutzerkreises führt, andererseits findet eine zunehmende Integration bislang isolierter Einzelanwendungen statt [BaS00].

Als erstes soll kurz erläutert werden, was CSCW eigentlich bedeutet und wie computergestützte Gruppenarbeit durch Software-Anwendungen unterstützt werden kann. Unter Kooperation wird das Zusammenarbeiten mehrerer Individuen bzw. Gruppen von Individuen mit dem Zweck, ein ganz bestimmtes Ergebnis anzustreben, verstanden. Ohne diese Form des Zusammenwirkens hätte sich die Gesellschaft nie entwickeln können, da das Einbringen verschiedener Fähigkeiten und Sichtweisen einen Arbeitsgegenstand erst formt. Das Ziel heutiger CSCW-Systeme ist es, die Zusammenarbeit von Menschen durch den

[30] Weitere Informationen: http://sea.symantec.com/content/product.cfm?productid=16

[31] Weitere Informationen: https://www.gotomypc.com

[32] Weitere Informationen: http://www.netopia.com/software/products/tb2/

[33] Weitere Informationen: http://www.positivenetworks.com

Einsatz von IuK-Technologien zu verbessern, d.h. sie effizienter und flexibler, aber auch humaner und sozialer zu gestalten [BaS00].

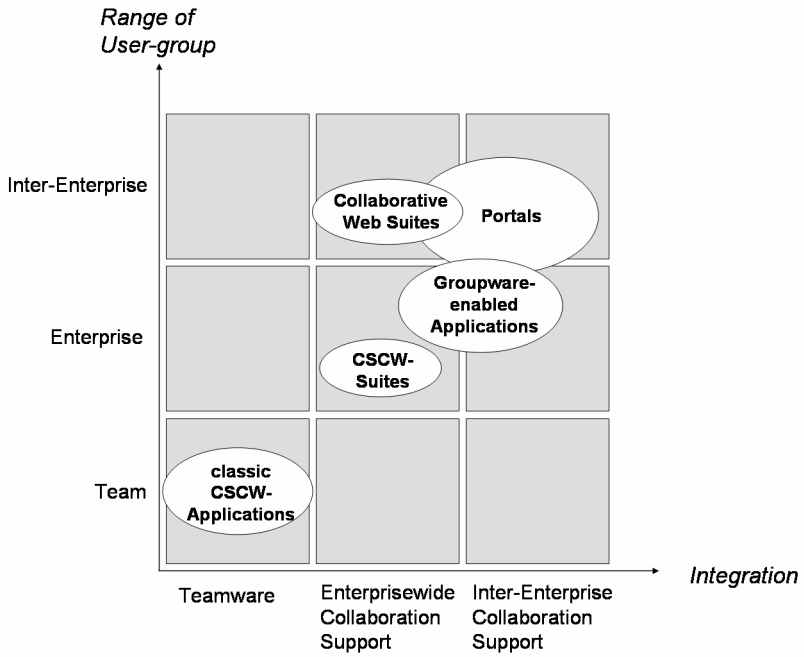

Abbildung 3-1: CSCW-Klassifikation nach Entwicklungsphasen [BaS00]

Als Grundlage jeder Kooperation muss allerdings eine funktionierende Kommunikation existieren. Zur Kommunikation gehören vor allem Erfahrungsaustausch und das Einbringen und Diskutieren von Ideen durch verschiedene Gruppenmitglieder. Darauf aufbauend kann eine Kooperation entwickelt werden, die ohne die Kommunikation und einer gewissen Kommunikationshygiene nicht bestehen könnte.

Zur technischen Unterstützung kooperativen Arbeitens wurden bisher vor allem zwei Kategorien von Anwendungen genutzt - Groupware und Workflowmanagementsysteme (WFMS). In den letzten Jahren haben sich auf der Basis der Internettechnologie verschiedenste Systeme zur Unterstützung der Kollaboration verteilter Gruppen etabliert, die unter der Rubrik webbasierte kooperative Systeme zusammengefasst und ebenfalls in diesem Kapitel vorgestellt werden.

3.2.1 Groupware

Computergestützte Werkzeuge, die speziell auf die Unterstützung der Gruppenarbeit bzw. des kooperativen Arbeitens ausgerichtet sind, werden allgemein unter dem Begriff „*Groupware*" oder auch „*Collaborative Support Systems*" zusammengefasst. Die Funktionalität, die von Groupware-Produkten bereitgestellt wird, zielt auf eine Zusammenarbeit in der Gruppe unabhängig von örtlichen und zeitlichen Gegebenheiten ab. Groupware kann Anwender mit benötigten Informationen zur Lösung von Aufgaben versorgen oder helfen, Entscheidungen zu treffen. Dabei unterstützen Groupware-Programme vor allem den flexiblen und schnellen Informationsfluss zwischen den am Problem beteiligten Gruppenmitgliedern und die aufgabenspezifische Gruppenzusammenstellung und -hierarchie.

Als Grundlage für die Kooperation besteht bei Groupware-Systemen oft das Konzept des gemeinsamen Materials. Dahinter steht die Grundidee, die für die Kooperation relevanten Informationen (Texte, Grafiken, Datensätze aus Datenbanken oder elektronische Doku-

mente) allen beteiligten Personen zugreifbar zu machen. Die Überbrückung von Zeit ist dabei ein wichtiger Funktionsbereich von Groupware. Die Motivation besteht darin, Personen die Zusammenarbeit unabhängig davon zu ermöglichen, ob sie gleichzeitig oder zu verschiedenen Zeiten an einem Problem arbeiten. Dies wird oft dadurch ermöglicht, dass die gemeinsame Aufgabe in Einzel- und Gemeinschaftsaktivitäten aufgeteilt wird: Einzelaktivitäten können von einem Bearbeiter allein erledigt werden; Gemeinschaftsaufgaben sind in der Gruppe zu bearbeiten. Beide basieren auf gemeinsamem Material und müssen untereinander koordiniert werden. Ebenso wichtig wie die Überbrückung von Zeit ist die Überbrückung von Entfernungen. In vielen dezentral organisierten Unternehmen oder im Rahmen von Telearbeit muss gerade für die örtlich voneinander entfernten Mitarbeiter eine spezielle Systemunterstützung bereitgestellt werden. Groupware-Systeme erlauben die Überbrückung der örtlichen Entfernungen oft durch synchrone Konferenzlösungen oder asynchronen Nachrichtenaustausch.

Traditionell enthalten Groupware-Lösungen folgende Kategorien von Werkzeugen:

- Asynchroner Nachrichtenaustausch: E-Mail-Systeme bieten die Basis zur elektronischen Nachrichten- und Dokumentenübermittlung. Darüber hinaus dienen flexible und skalierbare Instant-Messsaging-Systeme zum Austausch von Nachrichten zwischen den Gruppenmitgliedern.
- Kalender- und Terminverwaltung.
- Gruppen-Entscheidungssysteme.
- Synchrone Konferenzsysteme (Video, Audio, Daten)
- Asynchrone Konferenzsysteme (Foren, Blackboards, News)
- Dokumentenmanagement.
- Gemeinsame Arbeitsbereiche (*Shared Workspaces*): Diese Werkzeugklasse stellt gemeinsame Arbeitsbereiche für Teams zur Verfügung, in denen strukturierte und unstrukturierte Informationen in geeigneter Form gespeichert und bearbeitet werden können.
- (teilweise) Workflow-Unterstützung.
- Sicherheitsmechanismen - feingranulares hierarchisches Gruppen- und Rollen-Management.

Zurzeit gibt es drei große Unternehmen, die den Markt für Groupware-Systeme beherrschen (siehe Tabelle 3-1). Lotus Notes steht mit 29,5 Millionen weltweiten Nutzern bisher an der Spitze, verliert aber deutlich Marktanteile an den Konkurrenten Microsoft, der mit MS Exchange bisher 21,2 Millionen Nutzer an sich binden konnte. Als dritter „Global Player“ folgt Novell, das bisher 14,2 Millionen Nutzer seines Groupware-Systems Groupwise vorweisen kann [EiL04].

Anbieter	Produkt	Unterstützte Plattformen
Lotus/IBM	Notes (Client), Domino (Server)	Windows, OS/2, Unix, Linux
Microsoft	Outlook (Client), Exchange (Server, Client)	Windows
Novell	Groupwise	Novell Netware, Windows

Tabelle 3-1: Global Player im Groupware-Markt [EiL04]

3.2.1.1 LOTUS Notes und Domino

Lotus Notes ist ein weit verbreitetes Groupware-Programm, das vor allem von Großunternehmen zur innerbetrieblichen Kommunikation und Koordination genutzt wird. Dabei ermöglicht Lotus Notes als zentrale Schaltstelle den Zugriff auf alle wichtigen Informationen, egal ob sie aus einer relationalen Datenbank, einer E-Mail-Nachricht, einer Desktop-Anwendung oder aus dem Internet stammen. Mit dem *Organizer* wird ein Werkzeug zur Terminplanung und -koordinierung aller Mitarbeiter geliefert.

Der Lotus Domino Server und der dazugehörige Notes Designer bieten eine vollständige Plattform und Entwicklungsumgebung für Business-Applikationen im Intranet. Mit diesen Systemen lassen sich Lösungen entwickeln, die Groupware-, Sicherheits- und Workflow-Funktionen enthalten. Der Lotus Domino Server ist ein kombinierter Web-, Messaging- und Groupware-Server. Notes Designer vereinfacht die Erstellung und Verwaltung von interaktiven Internet- oder Intranet-Seiten und ermöglicht es Anwendern, auf eine sichere Art die vorhandenen Lotus-Notes-Applikationen auf jedem Client oder Webbrowser auszuführen. Lotus *cc:Mail* ist ein Messaging-System für kleine und mittlere Unternehmen, das leicht zu installieren und zu pflegen ist.

3.2.1.2 MICROSOFT Exchange und Outlook

Mit dem Exchange-Server bietet Microsoft eine Plattform mit umfangreichen Groupware-Funktionen wie *Messaging* und *Collaboration*. Exchange unterstützt zahlreiche Aktivitäten im Bereich der Zusammenarbeit wie Zeitplanungsfunktionen für Gruppen, Diskussionsgruppen und Teamordner. Mit integrierten Funktionen zum Indizieren und Suchen nach Inhalten können die Benutzer Daten problemlos und schnell suchen und gemeinsam nutzen.

Microsoft Outlook ist das Client-Programm zur Daten- und Informationsverwaltung, das Telearbeiter beim Organisieren und gemeinsamen Nutzen von Daten und bei der Kommunikation mit anderen Personen unterstützen kann. Folgende Funktionen sind mit Outlook nutzbar:

- Verwalten persönlicher oder geschäftlicher Daten wie E-Mail-Nachrichten, Termine, Kontakte, Aufgaben und Dateien sowie Verfolgen von Aktivitäten.
- Gemeinsames Nutzen von Daten in einer Arbeitsgruppe mit Hilfe von E-Mail, Zeitplanung für Arbeitsgruppen, öffentlichen Ordnern etc.
- Gemeinsames Nutzen von Daten mit anderen Office-Programmen, Suchen nach Office-Dateien von Outlook aus.
- Herstellen einer Verbindung und gemeinsames Nutzen von Daten mit dem World Wide Web.
- Nutzen der Programmieroptionen zum Anpassen von Outlook für Entwickler.

Mit *Outlook Web Access* (OWA) wird auch ein Zugang von Thin Clients über einen Webbrowser ermöglicht.

3.2.1.3 NOVELL Groupwise

Groupwise erweitert E-Mail um Werkzeuge für die Teamarbeit im Internet und Intranet. Es enthält Verwaltungsfunktionen für Dokumente und Grafiken und integriert Terminplanung, Aufgabenverwaltung, gemeinsame Ordner, Konferenzen, Workflow-Management und Internet-Zugriff nahtlos in einer universellen Mailbox. Das Dokumentenmanagement von Groupwise bietet Funktionen für Sicherheit, Management und Verwaltung von Daten, mit denen alle Informationen des Unternehmens sicher, aktuell und leicht zugänglich gemacht werden können. NDS (*Novell Directory Services*) bieten ein globales Verzeichnis und

einen zentralen Punkt für die Administration von Groupwise auch über Fernzugriff. Groupwise arbeitet auf allen wichtigen Servern und Clients, dies ermöglicht Teamarbeit auch in heterogenen Umgebungen. Groupwise unterstützt alle gängigen Internet-Browser und Internet-Protokolle, einschließlich SMTP/MIME, POP3, IMAP, LDAP und MAPI-kompatible E-Mail-Clients.

Weiterhin existieren noch eine Vielzahl von Open-Source-Projekten, die aber meist nur einen Teil der Goupware-Funktionalitäten abbilden können. In Kapitel 3.2.3 werden diese noch detaillierter beschrieben.

Folgende aktuelle Tendenzen im Groupware-Bereich sind zu beobachten:

- Integration der verschiedenen CSCW-Richtungen, vor allem von Groupware und WFMS, in einem System.
- Da vernetztes Arbeiten eine immer größere Rolle spielt, müssen die Systeme Internet/ Intranet-fähig gemacht werden. Dabei soll auch eine Verteilung der verschiedenen Komponenten des Systems auf mehrere Knoten ermöglicht werden, was die Leistungsfähigkeit und Verfügbarkeit der Systeme steigern kann.
- Neue Einsatzgebiete: virtuelle Unternehmen, mobile Computing, Wissensmanagement.
- Konzentrationsprozess der Anbieter.

Groupware bildet dabei vor allem schwach strukturierte Abläufe in Büroumgebungen ab und ist deshalb besonders für selbständiges Arbeiten geeignet, bei dem die Initiative zur Kommunikation und Kollaboration hauptsächlich vom Telearbeiter ausgeht. Groupware-Systeme bieten eine gute Grundlage für Telearbeit, wenn sie bereits durchgehend im Unternehmen vorhanden sind und genutzt werden.

3.2.2 Workflowmanagementsysteme

Klar strukturierte und immer wiederkehrende Abläufe in Unternehmen werden im Gegensatz dazu in Workflowmanagementsystemen (WFMS) abgebildet. Dabei übernimmt das System die Kontrolle über den Informationsfluss und die Steuerung der Kommunikation und Kooperation zwischen den beteiligten Bearbeitern. WFMS besitzen eine starke Aufgabenteilung sowie einen hohen Grad an Strukturierung.

Ein Workflowmanagementsystem wird von der *Workflow Management Coalition* (WFMC), einer weltweiten Initiative verschiedener Hersteller zur Standardisierung von WFMS, wie folgt charakterisiert:

> *„A system that completely defines, manages, and executes workflows through the execution of software whose order of execution is driven by a computer representation of the workflow logic.“* [WFM99]

Die grundlegende Idee des Workflow-Management ist die personen- und abteilungsübergreifende Betrachtung und Unterstützung komplexer Abläufe in Unternehmen, um durch eine Optimierung auf Prozessebene die Ausschöpfung von Reserven zu erreichen. Frühere Ansätze führten zwar meist eine Verbesserung der Ablauforganisation durch, konzentrierten sich beim Computereinsatz aber nur auf die Unterstützung und Automatisierung isolierter Aufgaben. Workflowmanagementsysteme (WFMS) betrachten dagegen den gesamten Prozess von seiner Initiierung bis zum Abschluss und streben einen höheren Automatisierungsgrad durch eine computergestützte Steuerung des gesamten Ablaufes an.

WFMS automatisieren die Ausführung von Geschäftsprozessen und ermöglichen es, während der Durchführung von Geschäftsprozessen zur richtigen Zeit Applikationen aufzurufen, menschliche Interaktionen zu erwarten und gleichzeitig die passenden Daten bereitzustellen. Im WFMS werden Geschäftsprozesse als Workflows abgebildet. Unter

einem Workflow wird die zeitlich-strukturelle Aneinanderreihung von einzelnen, zur Bearbeitung eines Gesamtvorganges notwendigen Teilaufgaben verstanden. Eine Folge von Teilaufgaben setzt sich aus einzelnen Aktivitäten zusammen, die durch Ereignisse ausgelöst und beendet werden. In der Regel sind Workflows organisationsweite, arbeitsteilige Prozesse, die sich aus verschiedenen Vorgangsschritten (Aktivitäten, Aufgaben) zusammensetzen und in die eine Vielzahl von Beteiligten involviert sind.

Ein wichtiger Begriff in diesem Zusammenhang ist der Begriff „Prozess“. Ein Prozess ist die logische, zeitliche Abfolge von Aktionen (oder Vorgängen), die für die Bearbeitung einer Aufgabe oder eines Objektes notwendig sind. Dem Prozess-Gedanken wird insbesondere durch eine detaillierte Analyse der Geschäftsvorfälle und deren schrittweise Integration in einem Prozessmodell Rechnung getragen. Es ist genau zu definieren, wo der Prozess beginnt und wo er endet. Dies kann man am Beispiel der Bearbeitung einer Rechnung erläutern. Der Prozess kann mit dem Rechnungseingang beginnen und sich über Rechnungsprüfung und Rechnungsbuchung bis zum Rechnungsausgleich erstrecken. Die in einem Workflow enthaltenen Teilschritte können gegebenenfalls parallel zueinander ausgeführt oder in weitere Teilschritte zerlegt werden.

Jeder Vorgangsschritt wird einem Bearbeiter zugeordnet, der dabei eine bestimmte Rolle einnimmt und verschiedene Ressourcen (Werkzeuge, Systemressourcen) benutzt. Eine Rolle steht in diesem Zusammenhang für einen Personenkreis, der berechtigt und fachlich befähigt ist, einen Vorgangsschritt auszuführen. Die Rolle repräsentiert dabei vielfach eine bestimmte Position in der Organisationsstruktur des jeweiligen Unternehmens. Die dabei benutzten Ressourcen sind beispielsweise Dokumente (Papier oder elektronisch), Anwendungssoftware und -werkzeuge sowie Informationen aus Datenbanken oder dem WWW. Für jeden konkreten Geschäftsprozess wird ein Vorgangsexemplar des betreffenden Vorgangstyps erzeugt, um dessen Bearbeitung zu steuern bzw. zu koordinieren. Während der Vorgangstyp den Geschäftsprozess auf einer abstrakten Ebene beschreibt, wird durch ein Vorgangsexemplar ein konkreter Geschäftsvorfall einschließlich der zur Bearbeitung benötigten Informationen repräsentiert.

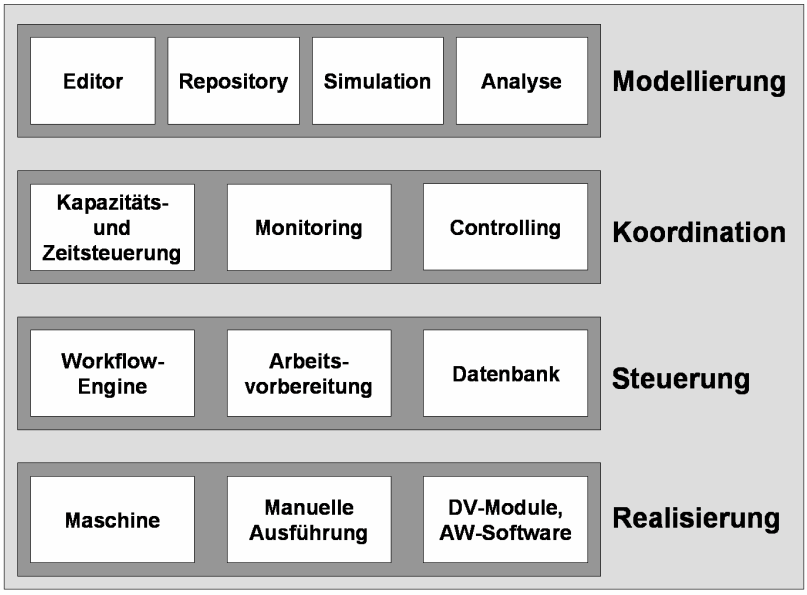

Abbildung 3-2: Architektur eines Workflowmanagementsystems [EiL04]

Die heutige Situation bei WFMS besteht in einer Vielzahl von Produkten mit unterschiedlicher Funktionalität, so dass an dieser Stelle nur ein grober Überblick zum Leistungsum-

fang von WFMS gegeben werden kann. Anhand des angebotenen Funktionsumfanges lassen sich WFMS wie folgt klassifizieren:

- *„Integrated Workflow Management Systems“* zeichnen sich durch eine geschlossene Produktstruktur aus. Diese Systeme enthalten alles, was für die Beschreibung der Vorgangstypen bis hin zur Benutzeroberfläche notwendig ist.
- „*Control- and Transport-oriented Systems“* besitzen keine Entwicklungsumgebung. Sie setzen Anwendungen voraus und übernehmen nur die Aufgabe des Bindegliedes zwischen den Anwendungen. Die Steuerung entlastet den Benutzer. Der Transport ermöglicht die Bereitstellung aller möglichen Daten.
- In den Fällen, in denen die Steuerungskomponente offen gelegt ist und in Form eines API's zugänglich ist, spricht man von der Gruppe der *„Flexible Development Systems“*.

Aus der koordinierten Prozesssteuerung ergeben sich eine Vielzahl von Vorteilen von WFMS. Ein Vorteil ist die genau definierte Ablaufstruktur und die Zuordnung von Ressourcen und Bearbeitern. Durch eine geeignete Modellierung der Geschäftsprozesse lässt sich der Ablauf im Unternehmen optimieren. Dies hilft den Unternehmen, Kosten zu senken und relativ schnell Entscheidungen fällen zu können. Allerdings lässt diese genau definierte Ablaufstruktur keine größeren Ausnahmen zu und ist damit sehr unflexibel. Weiterhin kann der aktuelle Status der Bearbeitung eines Vorganges stets und sehr schnell ermittelt werden, da alle dazu notwendigen Informationen im WFMS gespeichert sind. Ein Überblick über vorhandene Aufträge und ihren Zustand ist unter diesen Voraussetzungen ebenso möglich, was Vorteile bei der Kapazitätsplanung und Disposition bewirkt.

Die bei der Einführung von WMFS entstehenden Probleme erwachsen im Wesentlichen im organisatorischen und im sozialen Bereich. So ist die Überwachung der Arbeitsleistung jedes Bearbeiters prinzipiell möglich, da die Bearbeitungsdauer jedes Vorgangs bzw. die Anzahl der bearbeitenden Aufgaben auch nach deren Abschluss nachvollzogen werden kann. Aus diesem Grund sind entsprechende Maßnahmen zum Schutz personenbezogener Daten zu treffen.

WFMS steuern und koordinieren die modellierten Vorgänge und unterstützen den Benutzer bei der Durchführung der Aktivitäten. Dieser Ansatz eignet sich für stark strukturierte Organisationen und Unternehmen, in denen Aufgaben arbeitsteilig von mehreren Personen immer wieder in derselben Form erfüllt werden müssen und einen hohen Koordinationsaufwand erfordern. Für den Einsatz in kleinen und mittelständischen Unternehmen eignen sich WFMS wegen ihrer Mächtigkeit und des konzeptionellen Mehraufwandes bei der Analyse und Modellierung der einzelnen Geschäftsprozesse nicht immer. Ein weiterer Nachteil ist, dass heutige WFMS proprietäre Formate benutzen, wodurch ein Austausch zwischen verschiedenen Systemen nicht einfach möglich ist und damit die Anforderung an Interoperabilität nicht erfüllt werden kann.

3.2.3 Webbasierte kooperative Systeme

Webbasierte kooperative Systeme sind Anwendungen, welche die Zusammenarbeit mehrerer Menschen in einer Gruppe über die Web-Technologie ermöglichen. Web-Technologie bedeutet in diesem Zusammenhang grundsätzlich die Nutzung des *Hypertext Transfer Protocol* (HTTP). Die Definitionen webbasierter Dienste in der Literatur weichen in der Hinsicht stark voneinander ab, ob der Zugriff durch Client-Anwendungen oder nur über einen Standard-Browser erlaubt ist.

Um eine eindeutige Einteilung der verschiedenen Begriffe zu ermöglichen, erscheint eine Unterscheidung nach der Art der verwendeten Clientanwendung (Browser oder Installation entsprechender Clientsoftware) und nach der Übertragungsmöglichkeit durch standardmä-

ßig konfigurierte Firewalls sinnvoll. Damit entstehen drei große Gruppen von Diensten: client-, browser- und webbasiert. Die Einteilung ist in Abbildung 3-3 dargestellt.

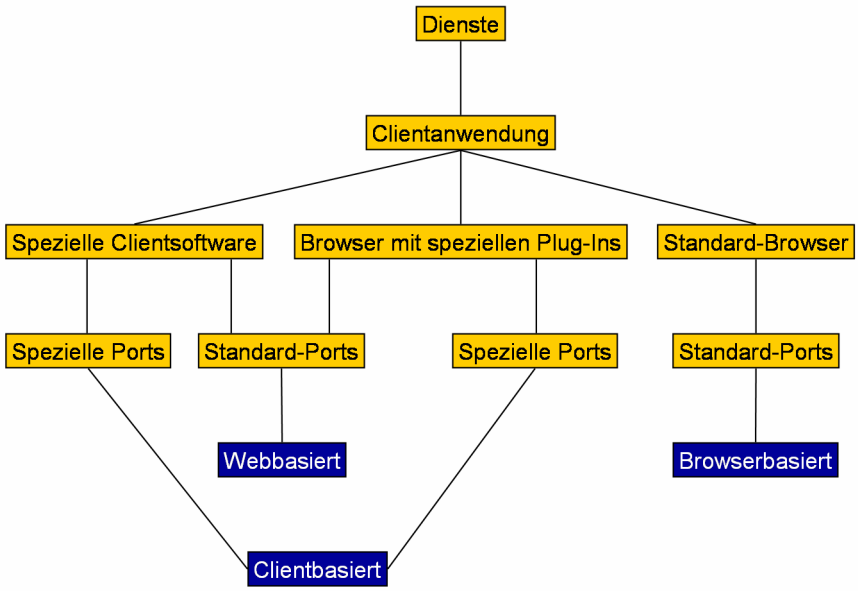

Abbildung 3-3: Einteilung webbasierter Dienste

Browserbasierte Dienste werden komplett ohne zusätzliche Komponenten in einem Standardbrowser ausgeführt. Es sollen folglich keine zusätzlichen Softwarekomponenten lokal auf dem Zielrechner installiert werden. Zusätzlich muss jede Kommunikation über den Port 80 bzw. 443, also mittels HTTP bzw. HTTPS, erfolgen. Unter Standardbrowser soll jeder beliebige, auf seiner entsprechenden Plattform laufende Browser verstanden werden. Beispiele hierfür wären Netscape, Internet Explorer, Opera oder Mozilla. Da heutzutage Java (nicht *Java Web Start*) als plattformunabhängige Erweiterung für die meisten Systeme verfügbar und installiert ist, soll es im Weiteren, ebenso wie *JavaScript*, nicht als spezielles Plug-In betrachtet werden. Hingegen sind Microsoft´s *ActiveX Controls* herstellerspezifisch und gelten hier als spezielles Plug-In. An Java-Applets muss die Zusatzanforderung gestellt werden, dass sie nur über die Standardports mit verteilten Java-Objekten auf anderen Servern in Verbindung stehen. Sollten sie dieser Anforderung nicht genügen, sind sie nach dieser Einteilung eher in die Gruppe der client-basierten Dienste einzuordnen.

Die Definition lässt schon das Problem der rein browser-basierten Dienste erahnen. CollabWorx Inc.[34] beschreibt es auf ihren Internetseiten [Col05] wie folgt: „Es ist nicht möglich, alle Protokolle in HTTP/S zu kapseln und reiche Funktionalität, Stabilität, Skalierbarkeit und Leistung von einer Echtzeitkommunikationsanwendung zu erwarten. Die beliebte Tendenz, alle Verbindungen auf der Firewall zu blockieren und verteilte Konnektivität nur über HTTP/S (bekannt als "Port 80 Dogma") zu erzwingen, ist äquivalent dazu, als ob die Gesamtheit der Arbeiten von IETF, ITU und W3C an Internetprotokollen ungültig gemacht werden würde.“

Somit dürfte klar sein, dass rein browserbasierte Dienste in ihrer Funktionalität beschränkt sind und zwangsläufig durch andere Lösungen, wie web- und clientbasierte Dienste,

[34] Hersteller von Software für Kollaboration und Kommunikation (http://www.webwisdom.com).

ergänzt werden müssen. Dennoch können browserbasierte Dienste eine Vielzahl der für Telearbeit notwendigen Funktionen erbringen.

Clientbasierte Dienste erfordern die Installation spezieller Client-Softwarekomponenten zur Verwirklichung der entsprechenden Funktionalität und stehen mit anderen Netzknoten über spezielle Ports in Verbindung. Damit ist die Integration in bestehende Sicherheitskonzepte nicht ganz unproblematisch. Es müssen Firewalls rekonfiguriert bzw. viele zusätzliche Ports geöffnet werden. Dadurch steigt die Zahl der Eintritts- und Angriffspunkte des Systems. Deshalb werden auch Dienste, die zwar über einen Browser, aber nur mit speziellen Plug-Ins und über zusätzliche Ports zu ereichen sind, zur Gruppe der clientbasierten Dienste gezählt.

Webbasierte Dienste vereinen client- und browser-basierte Dienste. Sie bieten oftmals einen autorisierten Zugang über eine Webseite zum System an. Über diese Webseite können eventuell nötige Softwarekomponenten installiert werden. Diese Komponenten können in Form von Plug-Ins vorliegen oder komplett eigenständige Anwendungen darstellen. Eines ist ihnen aber gemein: die Kommunikation mit den Server-Anwendungen erfolgt einheitlich über HTTP unter Einhaltung des "Port 80 Dogmas", also über Standardprotokolle.

Im Folgenden werden alle drei definierten Dienstarten berücksichtigt und zur Vereinfachung unter dem Begriff „webbasierte Dienste“ zusammengefasst.

So vielfältig wie die Definitionen sind auch die angebotenen Dienste und deren Funktionalität. Deshalb werden in Anlehnung an die in Kapitel 2.6 beschriebenen funktionalen Anforderungen Basisdienste und -funktionalitäten festgelegt, die ein webbasierter kooperativer Dienst erbringen muss, um für Telearbeit genutzt werden zu können.

Grob gesehen benötigen die meisten Telearbeiter Komponenten

- zur Kommunikation (sowohl synchron als auch asynchron),
- zur zentralen Speicherung und Verwaltung von Dateien,
- zum Verwalten von Terminen, Aufgaben und Adressen,
- zum gemeinsamen Arbeiten auf Dokumenten.

Zur asynchronen Kommunikation ist in den meisten Fällen die Möglichkeit des E-Mail-Versands ausreichend. Dennoch kommt auch dem Diskussionsforum und der Pinwand eine große Bedeutung zu. Ist eine synchrone Kommunikation zur schnellen Abstimmung notwendig, können Dienste wie Internet-Telefonie mittels Voice-Over-IP-Technologien oder Videokonferenz-Lösungen eingesetzt werden. Durch die Steigerung der Leistungsfähigkeit der verwendeten Codierungs- und Übertragungsalgorithmen, sowie der verwendeten Rechen- und Übertragungstechnik, haben diese Lösungen in den letzten Jahren einen regen Zuspruch erfahren.

In [RLB05] wurde eine Systematisierung von webbasierten Groupware-Anwendungen für die Kooperation in verteilten Projektteams und virtuellen Unternehmen vorgenommen und ein Überblick am Markt existierender Lösungen erarbeitet. Die zur Systematisierung aufgestellten technischen und sozio-technischen Kriterien können fast äquivalent auf den Telearbeitskontext abgebildet werden. Im Rahmen dieser Untersuchung wurden 74 kommerzielle und frei verfügbare Softwarelösungen aus den Bereichen Groupware, Content-Management-Systeme (CMS), Blog und Enterprise-CMS getestet und bewertet. CMS unterstützen primär die kooperative Erstellung und Verwaltung digitaler Inhalte. Blog-Systeme sind einfache CMS, die das Bearbeiten von Webseiten ohne HTML-Kenntnisse erlauben. Enterprise-CMS bieten im Gegensatz dazu einen sehr umfangreichen

Funktionsumfang und zusätzliche Tools zur Kollaboration und Kommunikation. Groupware-Systeme sind besonders auf die Unterstützung verteilter Gruppenarbeit ausgerichtet.

Im Ergebnis der Analyse konnte festgestellt werden, dass die Grenzen zwischen den Funktionsbereichen weiter verschwimmen und die Entwicklung hin zu ganzheitlichen Lösungen geht. Ein solches System ist z.B. *Tiki CMS/Groupware*, welches sowohl Funktionen klassischer CMS als auch erweiterte Groupware-Funktionen beinhaltet. Die Groupware-Anwendung *eGroupware* besticht durch ihre Unterstützung von Mehrsprachigkeit und die Synchronisation mit kommerziellen Groupware-Lösungen wie Microsoft Outlook. Die komplette Bewertung aller gestesteten Lösungen kann in [RLB05] nachgelesen werden.

Im Folgenden werden exemplarisch drei konkrete Softwarelösungen verschiedener Anbieter vorgestellt. Dabei wurden die oben genannten Standardfunktionen quasi als Mindestanforderung gesehen. Deshalb sollen an dieser Stelle Beispiele genügen, die (fast) alle genannten Funktionalitäten besitzen. Es gibt sicher noch eine Vielzahl spezifischer Lösungen, die einzelne Dienste effizienter verwirklichen, aber sie können meist nicht die Funktionalität eines kompletten „virtuellen Büros“ abbilden.

3.2.3.1 Basic Support for Cooperative Work *(BSCW)*

Das Projekt- und Dokumentenmanagementsystem BSCW (*Basic Support for Cooperative Work*) wird bereits seit 1995 durch die Gesellschaft für Mathematik und Datenverarbeitung (GMD), das Fraunhofer Institut FIT und die OrbiTeam Software GmbH entwickelt. BSCW ist eine vollständig webbasierte Groupware-Lösung, die neben Funktionen zum Dokumentenmanagement und zur Projektverwaltung auch ein ausgefeiltes Rollenkonzept anbietet. Folgende Funktionen können zur Kooperation und Koordination verteilter Projektteams genutzt werden:

- Einrichtung und Verwaltung von Arbeitsgruppen,
- Erstellen eines gemeinsamen Arbeitsbereichs für jede Projektgruppe,
- Hochladen von Dateien (Graphiken, Word-Dokumente, Präsentationen, Textdokumente, etc.) und Verweisen auf Web-Ressourcen,
- Versionsverwaltung bei der Bearbeitung von Dokumenten,
- Bewertung und Kommentierung der abgelegten Dokumente durch die anderen Projekt-Teilnehmer,
- Erstellen von Profilen für jeden Teilnehmer (mit Bild, Adresse, Homepage, etc.),
- Erstellen von Offline-Diskussionsforen für Arbeitsgruppen (ähnlich News-Servern),
- Verwendung eines gemeinsamen oder persönlichen Kalenders,
- Planung und Organisation von Meetings (auch virtuellen),
- unmittelbare Kommunikation mit Partnern, die gerade im gemeinsamen Arbeitsbereich aktiv sind,
- Einfach zu handhabende Schnittstellen zu externen Kommunikationstools (bspw. *CuSeeMe*-Videoconferencing oder Microsoft Netmeeting).

Der in Abbildung 3-4 dargestellte Screenshot der BSCW-Umgebung zeigt den so genannten Arbeitsbereich, in dem verschiedene Dokumente oder Verweise einem Projekt zugeordnet und in Verzeichnissen hierarchisch sortiert werden können. Durch das feingranulare Benutzer- und Rollenmanagement können für jeden Ordner oder jedes Dokument die Zugriffsrechte explizit festgelegt werden. Benutzer können verschiedenen Gruppen zugeordnet werden. BSCW bietet außerdem ein konfigurierbares Berichtssystem, das den Nutzer über alle Ereignisse in gemeinsamen Arbeitsbereichen informiert.

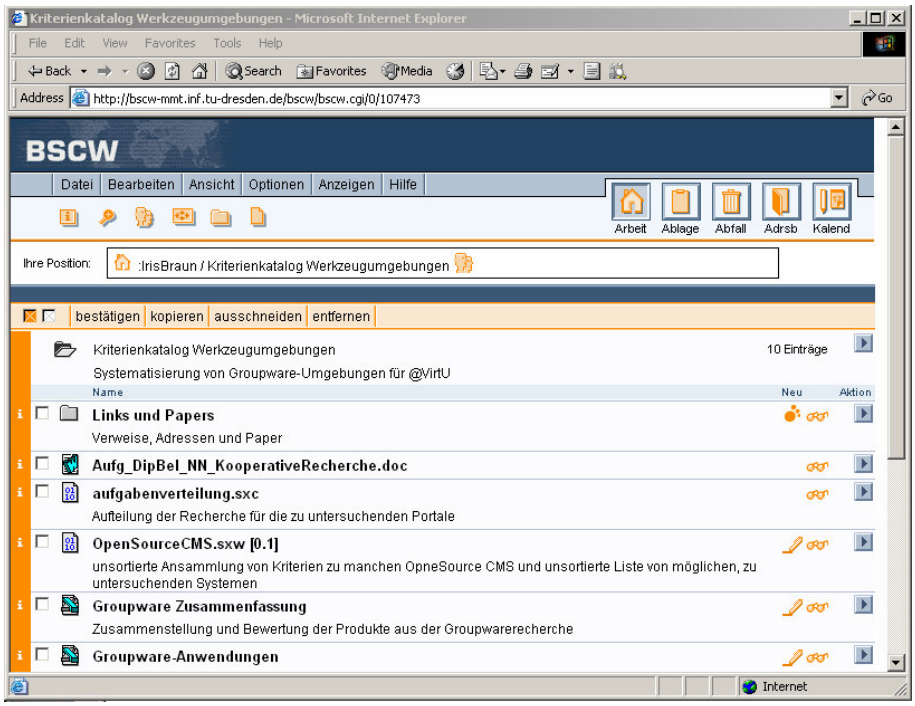

Abbildung 3-4: Oberfläche des BSCW-Systems

BSCW kann mit geringem Aufwand in (beinahe) jeden üblichen WWW-Server integriert werden und passt dadurch in jede vorhandene Netzwerkarchitektur. Der Zugang über das WWW sorgt für die Verfügbarkeit der Dokumente unabhängig von Zeit und Ort - ideal für die Bereitstellung von Dokumenten für externe Telearbeiter oder die Zusammenarbeit mit Kollegen. Durch den Thin-Client-Ansatz ermöglicht BSCW die Zusammenarbeit von beliebig großen Benutzergruppen, ohne eine Softwareinstallation bei den Endbenutzern vornehmen zu müssen. Da BSCW von nicht-kommerziellen Einrichtungen kostenlos genutzt werden kann, ist es im universitären und schulischen Bereich stark verbreitet.

3.2.3.2 Teamspace

Teamspace [Tea05] ist ein Portal für virtuelle Teamarbeit im Internet. Es ist modular aufgebaut und bietet außer Videokonferenzen und Voice-Over-IP alle genannten Standarddienste. Die notwendige Infrastruktur zur Kommunikation, Koordination und Kooperation wird online durch Teamspace bereitgestellt. Jede Arbeitsgruppe kann hierzu einen eigenen Teamraum anlegen und mit den notwendigen Funktionalitäten ausstatten. Der gestaltete Raum, der so genannte "*team space*", steht allen Gruppenmitgliedern sofort zur Verfügung und ist durch Benutzernamen, Passwort und SSL-Verschlüsselung geschützt. Zusätzlich gibt es noch einige praktische Module wie die Zeiterfassung. Diese ermöglicht die Erfassung des zeitlichen Aufwands von Arbeitspaketen und macht die anschließende Auswertung möglich. Dadurch können zukünftige Projekte besser geplant werden. Hierbei ist natürlich zu beachten, dass die Telearbeiter dadurch nicht überwacht werden sollen.

Die verfügbaren Module können wie folgt zusammengefasst werden:

- Terminplanung und Kalender,
- Aufgabenplanung und Projektmanagement,
- Dateiablage und Dokumentenmanagement,
- Bewertung und Ideenfindung,
- Adressverwaltung,
- Telegramme und E-Mail,

- Chat,
- Diskussionsforum,
- Pinwand und News,
- Zeiterfassung und Berichte.

Alle eben genannten Funktionen werden im Browser realisiert. Sie werden dabei sehr übersichtlich dargestellt (siehe Abbildung 3-5) und ermöglichen so die einfache Verwaltung und Kommunikation in Teams.

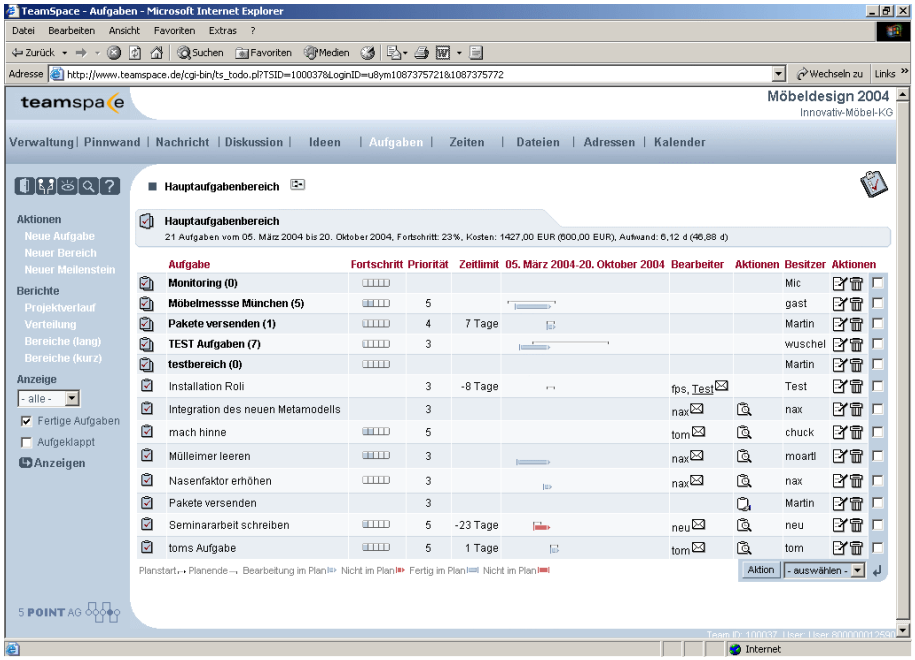

Abbildung 3-5: Teamspace-Oberfläche am Beispiel „Aufgabenplanung“

Beim Erwerb einer Enterprise-Server-Lizenz kann Teamspace im unternehmenseigenen Intranet sowie über Internet betrieben werden und ermöglicht dazu noch die Anpassung an das unternehmenseigene Design. Durch den Erwerb von Team-Lizenzen besteht aber auch die Möglichkeit der Nutzung der Online-Services von Teamspace. Der Nachteil bei Teamspace ist, dass das Dokumentenmanagement nur die Sperrung einer gerade benutzten Datei vorsieht und diese Datei allen anderen Nutzern während dieser Zeit nicht zur Verfügung steht.

3.2.3.3 Communispace

Communispace ist laut eigenen Angaben [Com05] einer der führenden Hersteller für web- bzw. browser-basierte Software und Dienste. Die Lösungen von Communispace betreffen einerseits das kooperative Arbeiten in Teams und andererseits die Unterstützung von Firma-Kunde-(B2C) bzw. Kunde-Kunde-Kommunikation (C2C). Beides wird unter dem Schlagwort „*Community Support*“ zusammengefasst. Wie auch bei Teamspace wird keine synchrone Kommunikation via Videokonferenz oder Voice-Over-IP unterstützt.

Die gesamte Lösung ist ebenfalls in Modulen organisiert, welche da wären:

- Diskussionsforum,
- Abstimmungen und Wahlen,
- Chat,
- Synchrone und asynchrone Brainstorming-Funktion (Ideenverwaltung),

- multimediale Galerie (z.B. entworfene Designs darbieten und Reaktionen erhalten),
- Referenz-Zentrum (Verwaltung von Benutzerfiles, Links, externe Webseiten usw.),
- Instant Messaging (schriftliche Kommunikation unter gerade aktiven Benutzern),
- Verwaltung von Benutzerprofilen,
- Suchfunktion (auch Volltext) auf den Unternehmensdatenbestand,
- Bücherladen (Beschreibungen von und Verknüpfungen zu Büchern oder anderen Artikeln, welche die Teilnehmer gegebenenfalls online kaufen können),
- Externe Aktivitäten (Integration von Drittanbietersoftware wie z.B. Centra),
- Berichte (Versendung von automatisch erstellten Berichten über Fortschritt der Arbeiten an verschiedene Personen wie Teammitglieder, Kunden oder Management),
- E-Mail-Benachrichtigung über Antworten oder andere neue Aktivitäten im Team.

Communispace stellt seine Anwendungen über das World Wide Web bereit. Die für die Anwender relevanten technischen Erfordernisse sind dabei auf Desktop-Ebene angesiedelt. Für die Nutzung der verschiedenen Funktionen reicht ein gewöhnlicher Browser, welcher allerdings SSL ab Version 2, Cookies, JavaScript und Java unterstützen muss. Zusätzliche Plug-Ins sind nicht notwendig.

Das Sicherheitslevel kann der Anwender nach den jeweiligen Erfordernissen selbst auswählen. Sollte eine Datenverschlüsselung notwendig sein, wird *Secure Socket Layer* (SSL) angeboten. Dafür wird für eine sichere Kommunikation über HTTPS mit zwei Schlüssellängen angeboten: 40 und 128Bit. Communispace gewährleistet Sicherheit beim verteilten Arbeiten generell durch die bestehende Multi-Tier-Struktur, bei welcher selbst die verschiedenen verteilten Dienste strikt von Diensten anderer Kunden getrennt sind. Generell sind alle Server physikalisch auf einer verteilten Anlage gesichert. Als Web-Server nutzt Communispace IIS 5.0 auf Windows 2000 laufend. Zusätzlich werden Microsoft´s *Actice Server Pages* ebenso wie COM-Objekte benutzt. Die Anwendung nutzt den Microsoft SQL Server 2000 als Daten-Repository.

3.2.3.4 Fazit

Zusammenfassend kann festgestellt werden, dass mittels reiner browser-basierter Dienste nur Standarddienste und einige ausgewählte Zusatzdienste zufrieden stellend realisierbar sind. Die Realisierung von Spezialdiensten in Browserumgebungen ist fast ausschließlich auf die Bereiche beschränkt, bei denen Informationen in Form von Text eingegeben werden können. Sobald es um graphische Eingaben geht, sind derartige Systeme mit browserbasierten Diensten nur mit sehr viel Aufwand umzusetzen.

Die meisten der betrachteten Systeme bieten aber eine Reihe von Kommunikations- und Kollaborationswerkzeugen und damit eine gute Unterstützung von entfernter und Gruppenarbeit. Für mittlere und kleine Unternehmen bieten sie eine kostengünstige Alternative zu kommerziellen Groupware- und Workflow-Systemen.

Da die vorgestellten Systeme alle auf eigenen proprietären Protokollen basieren und keine Standard-Schnittstellen zur Integration weiterer Dienste zur Verfügung stellen, erfüllen sie das Kriterium der Erweiter- und Skalierbarkeit und damit der Universalität nicht. Sie stellen aber zum Einstieg eine kostengünstige Alternative gegenüber der Entwicklung einer unternehmenseigenen Lösung dar.

3.3 Integrationslösungen

Im Rahmen von Business-Reengineering-Maßnahmen arbeiten viele Unternehmen derzeit daran, konventionelle, papierorientierte Arbeitsprozesse in den Unternehmen durch flexible, webbasierte Plattformen zur Unterstützung von Geschäftsprozessen innerhalb und außerhalb der Unternehmen zu ersetzen. Neue Technologien auf Basis von offenen Web-Standards ermöglichen es, die Integration von unternehmensinternen und Systemen der Geschäftspartner zu verwirklichen. Geschieht dies auf der Ebene der Endbenutzerinteraktionen spricht man von *Enterprise Portals*, wird eine Anwendungsprozessintegration durchgeführt von *Enterprise Application Integration* (EAI). Beide Technologien entwickeln sich zunehmend zu integralen Bausteinen moderner Systemarchitekturen und sollen deshalb in den folgenden Abschnitten näher untersucht und beschrieben werden.

3.3.1 Enterprise Portale

Aus dem Lateinischen stammend, wird Portal vom Wort "porta" - die Pforte, abgeleitet. Das wesentlichste Merkmal einer Pforte ist die Tatsache, dass sie den Zugang zu etwas darstellt. Portale bilden so genannte „*single points of access*“, zentrale Zugangspunkte für dezentrale Informationssysteme. Anfänglich dienten Portale vorwiegend zur zentralen Bereitstellung von Informationen für bestimmte Nutzergruppen und bildeten im klassischen Sinne Einstiegsseiten in das weltweite Internet. Ausgestattet mit Suchfunktionen unterstützen sie den Nutzer beim Finden und Durchsuchen der angebotenen Seiten. Somit können Suchmaschinen wie Yahoo oder Google zu den klassischen Portalen gezählt werden, denn sie bieten einen zentralen Zugangspunkt zu einer Fülle von bereitgestellten Informationen.

Mit der rasanten Entwicklung des WWW und der dazugehörigen webbasierten Technologien kam es bald zur Integration neuer Funktionalitäten wie z.B. Kalendern, E-Mail-Clients und persönlichen Farbschemaeinstellungen. Diese Entwicklung gipfelte in so genannten "personalisierten“ Portalen, bei denen sich der Benutzer mit einer persönlichen Kennung und einem Passwort anmeldet und daraufhin das Portal speziell auf seine Ansprüche und Einstellungen zugeschnitten angezeigt wird. Somit kann der Nutzer die für ihn wichtigen Informationen, Dokumente und Funktionalitäten im Portal nach seinen Wünschen anordnen und bei jeder Anmeldung auf diese personalisierte Anordnung zugreifen. In den letzten Jahren entwickelten sich Portale zu virtuellen, elektronischen Plattformen, über die eine einheitliche Abbildung unternehmensinterner oder externer Informations-, Handels- bzw. handelsbegleitender Prozesse schnell und effizient realisiert werden kann [GuÖ03]. Diese Entwicklung führte zu einer speziellen Art von Portalen - den Unternehmensportalen oder auch *Enterprise Portals* genannt. Während öffentliche Portale wie AOL oder Yahoo eine breite Palette von Benutzern unterstützen, richten sich Unternehmensportale an die Benutzer eines einzelnen Unternehmens oder seiner Partnerunternehmen.

Der Begriff Unternehmensportal wird in der Literatur sehr unterschiedlich interpretiert. Die Eingrenzung reicht von engeren Formen, die nur den persönlichen Zugang von Mitarbeitern zum Informationsangebot eines Unternehmens verstehen (u.a. [Öst03]), bis zu weitläufigen Interpretationen (u.a. [GuÖ03]), die alle unternehmensweiten und -übergreifenden Marktplätze synonym zu E-Business-Portalen als Enterprise Portals bezeichnen. Die Entwicklung zeigt, dass die Grenzen immer weiter verschwimmen, da die Personalisierungsfunktionen moderner Portallösungen eine Differenzierung verschiedener Nutzergruppen und der ihnen zur Verfügung gestellten Informationen soweit ermöglichen, dass alle verschiedenen Anforderungen mit einer einzigen Lösung abgedeckt werden können. Deshalb wird im Folgenden die allgemeine Definition zugrunde gelegt und synonym

für „Unternehmensportal“ verwendet, um die grundsätzliche Funktionsweise der technischen Lösungen zu erörtern. Für eine engere Definition des Begriffs wird das Synonym „Mitarbeiterportal“ eingeführt, was eine deutlich bessere Abgrenzung ermöglicht.

Oft vereinen Unternehmensportale klassische Portalfunktionen wie die Bereitstellung von allgemeinen Informationen und Dokumenten für gewöhnliche Interessenten und gleichzeitig die Möglichkeiten des personalisierten Zugangs zu speziellen Unternehmensdaten. Diese Personalisierung kann aber noch einen Schritt weiter gehen. Die Unternehmensportale unterscheiden bei der Bereitstellung der Informationen zwischen Kunden (*Business to Customer*, B2C), anderen Unternehmen (*Business to Business*, B2B) und Mitarbeitern (*Business to Employee*, B2E). Dabei haben alle Unternehmensportalarten das Ziel, die zielgruppenspezifische Darstellung von Geschäftsprozessen zu unterstützen bzw. deren Abwicklung zu optimieren und notwendige Bearbeitungswerkzeuge bereitzustellen. „Die nutzerorientierten Prozesse werden dabei auf Basis der vorhandenen unternehmensinternen Prozesse und Anwendungen realisiert und bilden für den Nutzer eine zielgruppenorientierte Schnittstelle." [GuÖ03]

Beispielsweise ist es den Zuliefern der Volkswagen AG durch das Unternehmensportal *B2B Supplier Platform*[35] stets möglich, auf aktuelle Daten aus den VW-Systemen zuzugreifen und so den auf weite Sicht notwendigen Materialbedarf abzuschätzen und die eigene Produktion entsprechend zu steuern. Auf diese Weise werden nach Angaben der Volkswagen AG *B2B Supplier Platform* über 5500 Lieferanten erreicht[36]. Neben den Angeboten an Kunden und Lieferanten bietet die Volkswagen AG auch ein Portalsystem für ihre eigene Vertriebsstruktur an. In diesem *PartnerNet* werden 2.500 Autohändler mit ca. 50.000 Nutzern an ein Extranet angeschlossen [Web03]. Ein Ziel dieses Projektes war, Probleme durch ungleichmäßig verteilte Informationen im dezentralen Vertriebsnetz zu vermeiden. So werden z.B. Produktschulungen, Tipps zum Gebrauchtwagenverkauf und CI-Richtlinien den entsprechenden Mitarbeitern des Händlers zur Verfügung gestellt.

Die Funktionalität des Systems erschöpft sich jedoch nicht in der Verteilung von Informationen aus dem Konzern an die Händler. Letztere stehen in direktem Kontakt mit den Kunden des Unternehmens und müssen daher zuverlässig Auskunft geben können und gegebenenfalls auch Feedback an die VW AG senden können. So ist ein wichtiger Schwerpunkt der Entwicklung die Möglichkeit der Händler, Kundenanfragen, Produktprobleme und Erfahrungen zuverlässig und zeitnah an die zuständigen Mitarbeiter des Konzerns weiterleiten zu können (vgl. [Web03]). Zur Bereitstellung dieser Arbeitsumgebung wurden 21 separate Anwendungen in das Portal integriert. So finden sich neben dem *PartnerNet*, *PartnerShop* und *iTV* auch das *ServiceNet* und eine Applikation zur direkten Kommunikation mit den Systembetreuern (*HotlineChannel*).

Trotz des großen Aufwands beim Betrieb einer komplexen Portallösung, können positive Ziele für das Unternehmen realisiert werden. An diesem Beispiel zeigt sich, dass die Händler effektiver arbeiten können und stärker an den Konzern gebunden werden. Nach einer Umfrage beurteilen 98,8 Prozent der Nutzer das etablierte PartnerNet mit „gut“ bis „sehr gut“ [Web03] und belegen damit, dass durch den Einsatz von Portalen, die Zufriedenheit von Kunden und Partnern gesteigert werden kann.

Um Telearbeit zu unterstützen, können die bereits erwähnten Mitarbeiterportale zum Einsatz kommen. Nun ist diese Idee nicht ganz neu. So genannte *Intranet*-Lösungen können als Vorstufe für Mitarbeiterportale [GuÖ03] angesehen werden. Zur Abgrenzung werden im Folgenden die Begriffe „Intranet“ und „Portal“ etwas genauer definiert. Ein Intranet

[35] http://www.vwgroupsupply.com.

[36] Stand: 27.01.2004.

stellt Informationen auf Basis statischer HTML-Seiten bzw. eines *Content Management Systems* (CMS) bereit, aber nur an die berechtigten Mitarbeiter mit Zugang zum Intranet. Einzelne webbasierte Anwendungen sind aus dem Intranet über eine Verlinkung erreichbar.

Die Defizite werden durch die verwendeten Technologien rasch deutlich: Intranets dienen primär der Informationsverteilung und tragen damit nur am Rande zur Optimierung von Prozessen bei. Eingebundene Anwendungen sind untereinander nicht logisch vernetzt. Die Folge ist eine Anwendungsorientierung statt der notwendigen Orientierung an den Prozessen der Nutzer. Die fehlende Verbindung der Anwendungen geht aber auch mit der Notwendigkeit immer wiederkehrender Benutzeranmeldungen beim Arbeiten mit verschiedenen Anwendungen einher. Benutzernamen und Passwörter sind nicht unternehmensweit für alle Anwendungen, auf die ein Benutzer Zugriffsrechte besitzt, einheitlich.

Ein Portal ist eine Applikation, welche basierend auf Web-Technologien einen zentralen Zugriff auf personalisierte Inhalte sowie bedarfsgerecht auf Prozesse bereitstellt [GuÖ03]. Charakterisierend für Portale sind die Verknüpfung und der Datenaustausch zwischen verschiedenen Anwendungen über eine Plattform. Sie stellen die Schnittstelle zwischen Mitarbeitern und Prozessen bzw. Systemen der Unternehmung dar. Dabei können sie in der höchsten Form als "*Workplace*" das gesamte Arbeitsumfeld, d.h. alle notwendigen Anwendungen, bereitstellen. Eine manuelle Anmeldung bei den im Portal integrierten Anwendungen ist nicht mehr notwendig. Die für die verschiedenen Anwendungen notwendigen Zugriffberechtigungen werden durch ein Verfahren namens „*Single-Sign-On*“ realisiert. Dazu werden alle notwendigen Zugangsinformationen beim Betreten des Portals bereitgestellt und entsprechend den Anwendungen zur Verfügung gestellt. Portale bieten somit die Möglichkeit, übergreifende Prozesse und die Zusammenarbeit innerhalb heterogener Gruppen zu unterstützen.

Da Portale im Allgemeinen im Internet zur Verfügung stehen, können sie auch den Zugang mobiler Mitarbeiter zum Unternehmen ermöglichen. Es kann unabhängig von Ort und Zeit auf alle relevanten Daten zugegriffen und alle benötigten Applikationen benutzt werden. Das ist ganz klar im Interesse von mobilen oder On-Site-Telearbeitern. Diese können so auf E-Mail-Dienste, Adressverwaltungen, Diskussionsforen, Kalender, Unternehmensinformationen, Berichte etc. zugreifen. Außerdem besteht durch die Integration von Dokumentenmanagementsystemen (DMS) die Möglichkeit, eigene Dokumente zentral zu verwalten und anderen Mitarbeitern oder Kunden direkt zur Verfügung zu stellen.

3.3.2 Enterprise Application Integration

Hinter dem Begriff *Enterprise Application Integration* (EAI) verbirgt sich eine Vielzahl verschiedener Interpretationen. Ein gewisser Konsens herrscht in der Auffassung, dass es sich bei EAI um eine Strategie zur Integration von heterogenen Anwendungssystemen in eine einheitliche IT-Architektur eines Unternehmens handelt. Über die begriffliche Abgrenzung herrscht jedoch Uneinigkeit. Während einige Autoren EAI von Integrationstechniken wie Funktionsintegration (RPC) oder Datenintegration (RDA) abgrenzen [Wik05], werden diese Vorgehensweisen bei anderen Autoren mit in die Definition eingeschlossen (u. a. [DLP+02]). Einige Quellen (u.a. [ScW02]) zählen auch die benutzerorientierte Integration auf Präsentationsebene durch Portale zum Gebiet der EAI, andere trennen EAI- und Portal-Ansätze.

Ursprung der EAI ist der Wunsch, die Anwendungen eines Unternehmens derart zu verknüpfen, dass Geschäftsprozesse unabhängig von Systemgrenzen ablaufen können. EAI umfasst dabei die Planung, die Methoden und die Software, um heterogene, autonome Anwendungssysteme zu integrieren [Wik05]. Die Mittel, mit denen das realisiert werden kann, sind dabei sehr vielfältig. Eine nahe liegende Möglichkeit wäre es, sämtliche Daten

und Anwendungen in einem neuen System zu vereinen und somit sämtliche Geschäftsprozesse auf eine neue Software und Arbeitsweise umzustellen. Dieses Vorgehen mag in einigen Fällen sinnvoll sein, jedoch führt eine komplette Umstellung der Prozesse auch zur Einführung neuer Fehler und Probleme. Neu entwickelte Software ist selten so stabil und funktionssicher wie eine seit Jahren gewartete und bereinigte „Alt-Software"; es kann folglich sinnvoll sein, darauf zurückzugreifen. Deshalb ist ein wichtiger Grundsatz bei EAI die Integration vorhandener Anwendungen, so genannter *Legacy*-Systeme.

Ein weiterer Ansatz zur Vermeidung von Inkonsistenzen bei der Zusammenarbeit verschiedener Unternehmensanwendungen kann es sein, jede einzelne Anwendung mit Schnittstellen zu den anderen Systemen des Unternehmens auszustatten und somit sicher zu stellen, dass jede Datenänderung in allen an der Transaktion beteiligten Subsystemen durchgeführt wird. Diese Lösung zieht einen nicht unerheblichen Aufwand an Entwicklungsarbeit nach sich, da `n` Systeme `n * (n - 1)` Schnittstellen besitzen, die alle entwickelt und getestet werden müssen. Soll später ein weiteres System integriert werden, ist erneut ein extrem hoher Arbeitsaufwand notwendig, da in jedem der Systeme Änderungen vorgenommen werden müssen. Günstiger kann daher die Verwendung einer so genannten *Middleware* sein. Hierbei wird die Anzahl der Schnittstellen auf `2n` reduziert, was die Komplexität der Arbeiten deutlich verringert. Auch bei der späteren Integration eines weiteren Systems ist nur noch die Schnittstelle zwischen diesem und der Middleware zu erstellen oder anzupassen. Mit der Middleware-Technologie werden bestehende Anwendungen in einer Zugriffsschale gekapselt, die standardisierte Schnittstellen für den Nachrichtenaustausch mit anderen Anwendungen zur Verfügung stellt.

Die Verbindung der Systeme kann auf unterschiedlichen Ebenen erfolgen. Zur Kopplung in der Ausführungsebene werden *Remote Procedure Calls* (RPC) verwendet. Diese entfernten Funktionsaufrufe entsprechen im Wesentlichen der Anbindung von Software durch dynamisches Linken (Bibliotheken), jedoch ist bei RPC der Aufruf auch über ein Netzwerk (z. B. das Internet) möglich. Ein wesentlicher Nachteil dieser Vorgehensweise ist die erzwungene Synchronität der Aktionen - der aufrufende Prozess bleibt blockiert, bis der Server die Anforderung bearbeitet und geantwortet hat. Zudem ist bei Ausfall der Verbindung zwischen den Kommunikationspartnern keines der Systeme mehr funktionsfähig. Da die RPCs im Allgemeinen fest in den Programmcode eingebettet werden, erzwingen Änderungen an einem System oft auch die Neuerstellung aller Kommunikationspartner.

Eine Alternative zur Kopplung von Systemen auf Ausführungsebene ist der *Remote Data Access* (RDA). Hierbei kann auf die Daten aller Anwendungen unabhängig von der eigentlichen Anwendung und der zugrunde liegenden Datenbank zugegriffen werden. Die Abfrage der Daten erfolgt mittels einer standardisierten Sprache wie SQL[37]. Das wohl bekannteste Beispiel von RDA ist die ODBC[38]-Schnittstelle von Microsoft. Bei diesem Vorgehen werden jedoch ausschließlich die Daten der Systeme zusammengefasst; die Integration der Funktionalität ist zwar prinzipiell über so genannte *Stored Procedures* innerhalb der Datenbank möglich. Da diese jedoch stark von der zugrunde liegenden Datenbanksoftware abhängen, erschwert dies die Migration des Systems bedeutend.

Der Datenintegration steht die Prozessintegration gegenüber. Hierbei werden in einem ersten Schritt die Geschäftsprozesse im Unternehmen analysiert. Im Rahmen dieser Modellierung kristallisieren sich bestimmte Aufgabenstellungen heraus, die auf verschiedene oder gar alle Systeme des Unternehmens sowie die vorhandenen Daten zugreifen. Die im Laufe

[37] SQL (Structured Query Language) ist eine strukturierte Abfragesprache für relationale Datenbanken.

[38] ODBC (Open DataBase Connectivity) ist eine standardisierte Programmierschnittstelle, die es erlaubt, Anwendungen unabhängig von der verwendeten Datenbank zu entwickeln.

der Jahrzehnte gewachsene Systemlandschaft der Unternehmen ist aber typischerweise durch die vertikale Trennung der Applikationen entsprechend der Funktionsbereiche eines Unternehmens gekennzeichnet. Die Integration der Geschäftsprozesse erfolgt dagegen entlang einer horizontalen Verbindung der verschiedenen Funktionsbereiche. Sie bricht die Abteilungsgrenzen zwischen den vertikalen Applikationen auf. Die Integration der Anwendungen entlang der Geschäftsprozesse ermöglicht auch eine bessere Nutzung vorhandener Daten, z.B. konsolidierter Kundendaten, deren abteilungsübergreifende Kenntnis erst den sinnvollen Einsatz eines *Customer Relationship Managementsystems* (CRM) ermöglicht. Am Ende der Prozessintegration entsteht im Idealfall ein System, bei dem weitgehend automatisiert ablaufende Prozesse ihre realen Gegenstücke im Unternehmensalltag komplett abbilden und alle beteiligten Anwendungen im Unternehmen automatisch in den Prozess integriert werden.

Eine weitere Option zur Verknüpfung von Systemen ist die lose Kopplung von Anwendungen über den Austausch von standardisierten Nachrichten – also die Nutzung so genannter *Message oriented Middleware* (MOM). Will eine Softwarekomponente A mit einer Softwarekomponente B kommunizieren, sendet sie eine Anfrage an B. Die Nachricht wird von der Middleware-Komponente über ein Netzwerk weitergereicht und wenn notwendig in Warteschlangen, so genannten *Message queues,* zwischengespeichert. Dabei werden in der Regel gebräuchliche Netzwerkprotokolle wie HTTP verwendet. Auf der Empfängerseite setzt die Middleware die Anforderung in einen Funktionsaufruf an die Software B um. Gegebenenfalls leitet sie die Antwort der Komponente B an Komponente A auf demselben Weg zurück. Ein Beispiel für MOM sind Web Services, die auf dem Austausch von SOAP-Nachrichten beruhen.

Ein weiterer Begriff, der oft im Zusammenhang mit EAI verwendet wird, ist „*Enterprise Service Bus*". Dieses Konzept stellt eine Infrastruktur zur Verknüpfung von heterogenen Anwendungslandschaften eines Unternehmens zur Verfügung und bildet dabei gewissermaßen das Rückgrat der Unternehmens-IT. Dabei arbeitet ein ESB im Gegensatz zu herkömmlicher RPC-basierter Middleware mit dokumentenorientierten Verarbeitungsmodellen. Dabei kommunizieren die Anwendungen nicht direkt miteinander sondern über die *Message Queues* des ESB. Deshalb stellt ein ESB eine Art *Message oriented Middleware* dar. Er garantiert die einmalige Zustellung von Nachrichten an den richtigen Empfänger, ermöglicht die Verarbeitung unter Transaktionskontrolle und unterstützt verschiedene Messaging-Modelle. Bei einer Punkt-zu-Punkt-Kommunikation wird die Nachricht von einem Sender an genau einen Empfänger gesendet. Beim „*Publish and Subscribe*"-Ansatz wird eine Nachricht von einem Sender produziert, aber gegebenenfalls von mehreren Empfängern gelesen, die ein so genanntes „Topic" abboniert haben [Deg05].

Das Ziel aller EAI-Konzepte ist die Kopplung bisher voneinander isolierter heterogener Anwendungen. Dabei soll die Komplexität der Schnittstellen so gering wie möglich gehalten werden, was durch die Schaffung einer Integrationsschicht, der so genannten Middleware, realisiert werden kann. Die vorgestellten Middleware-Technologien beinhalten durchgehend die Möglichkeit, von den zugrunde liegenden Betriebssystem- und Netwerkschichten zu abstrahieren, und ermöglichen so die plattformunabhängige Kopplung heterogener Anwendungen. Nachteil der meisten EAI-Lösungen ist die höhere Komplexität des Gesamtsystems und die damit verbundene aufwendigere Wartung der Schnittstellen sowie die gestiegene Ausfallwahrscheinlichkeit des Gesamtsystems. Die meisten Konzepte und EAI-Plattformen ermöglichen zwar eine prozessorientierte Integration der Anwendungen, aber selten eine Integration der Benutzerschnittstellen in eine einheitliche Oberfläche und damit einen einfachen Zugang zu den komplexen Lösungen.

3.4 Fazit

Mit Hilfe von Terminaldiensten wird das entfernte Arbeiten sehr einfach und unkompliziert unterstützt. Man erhält direkten Zugriff auf den entfernten Arbeitsrechner und kann dort alle Anwendungen und Funktionen nutzen, die man auch im „normalen“ Büroalltag benutzt. Der Einarbeitungsaufwand ist dadurch sehr gering und auch die Installationskosten sind fast zu vernachlässigen, da die meisten modernen Betriebssysteme solche Dienste bereits integrieren. Nachteil ist die schlechte Qualität der graphischen Darstellung und die Verzögerung nach Eingaben, die durch die Netzwerkverbindung entsteht. Um einigermaßen zügig arbeiten zu können, muss man mindestens über einen DSL-Zugang verfügen. Mit ISDN oder gar Modem ist ein effektives Arbeiten kaum möglich. Benötigt man Anwendungen mit graphisch anspruchsvolleren Oberflächen, gibt es einige Einschränkungen durch die gedrosselte Farbtiefe und Auflösung. Weiterhin bilden Terminalzugänge immer ein hohes Sicherheitsrisiko, da sie den uneingeschränkten Zugriff auf den Rechner erlauben. Deshalb sollte auf jeden Fall nur über eine VPN-Verbindung mit dem Terminalserver kommuniziert werden. Dies zieht wiederum einen gewissen Installations- und Wartungsaufwand nach sich.

Groupware-Systeme unterstützen vor allem die Arbeit in verteilten Teams. Sie stellen den Nutzern eine Reihe von Kommunikations- und Kollaborationswerkzeugen zur Verfügung. Im Kontext von Telearbeit können diese Werkzeuge sehr hilfreich sein, um die Kommunikation mit den Kollegen und Kunden aufrecht zu erhalten und gemeinsam an Dokumenten zu arbeiten. Grundvoraussetzung ist natürlich, dass diese Groupware auch im restlichen Unternehmen eingesetzt wird, damit keine Medienbrüche in der Kommunikation entstehen. Der Nachteil von Groupware-Systemen ist, dass sie keine konkreten Arbeitsabläufe oder Geschäftsprozesse unterstützen, sondern nur die Werkzeuge zur Kommunikation und Kollaboration zur Verfügung stellen. Heutige Groupware-Lösungen sind meist geschlossene Systeme mit proprietären Datenformaten, die eine Integration weiterer Dienste und Werkzeuge nicht zulassen. Sie bieten meist eine sehr große und unübersichtliche Palette an Funktionen und lassen sich somit nicht flexibel an die Bedürfnisse des einzelnen Telearbeiters und seines Unternehmens anpassen. Kommerzielle Systeme sind außerdem sehr kostenintensiv in der Beschaffung und Wartung, was für kleinere Unternehmen oft ein Ausschlusskriterium darstellt.

Workflowmanagementsysteme bilden im Gegensatz zu Groupware die Geschäftsprozesse und Arbeitsabläufe im Unternehmen ab. Sie übernehmen dabei sowohl den Kontrollfluss der Kommunikation zwischen den beteiligten Mitarbeitern als auch die Steuerung der benötigten Anwendungen. Nachteil dieser Systeme ist, dass die Abläufe im Vorfeld modelliert werden müssen und nicht flexibel oder ad-hoc durch den Telearbeiter angepasst werden können. Sowohl Groupware-Systeme als auch WFMS bieten eine Reihe von Funktionen, die auch über webbasierte Clients genutzt werden können. Die flexible Integration aller benötigten Dienste für Telearbeit in einer einheitlichen Umgebung ist damit aber nicht realisierbar, da sie auf proprietären Protokollen aufbauen und nicht alle Dienste über eine standardisierte Schnittstelle zur Verfügung stellen.

Webbasierte kooperative Systeme stellen eine kostengünstige Alternative zu kommerziellen Groupware- und Workflow-Lösungen dar. Sie bieten eine Vielzahl von Werkzeugen zur Unterstützung der Kommunikation und Kollaboration in Arbeitsgruppen und sind somit für den Einsatz im Rahmen von Telearbeit gut geeignet. Ein wichtiger Pluspunkt ist die Umsetzung des webbasierten Thin-Client-Ansatzes und damit die Erreichbarkeit von überall ohne großen Installationsaufwand. Nachteil dieser Lösungen ist aber, dass sie nicht alle im „virtuellen Büro“ benötigten Anwendungen und Werkzeuge zur Verfügung stellen

und durch ihre proprietären Datenformate und fehlenden Schnittstellen nicht flexibel erweitert werden können.

Auch die erwähnten Integrationslösungen bieten eine gute Grundlage für Telearbeit, weil sie die Auslagerung der Geschäftprozesse des Unternehmens optimal unterstützen. Inzwischen geht der Trend zur Kombination von EAI und Portalen in einer einzigen Lösung. Einige Anbieter haben diese Entwicklung und deren Marktchancen bereits erkannt und offerieren eigene Produktkombinationen. Neben SAP mit *mySAP* bietet beispielsweise auch IBM mit IBM *WebSphere Portal* eine solche Kombination an. Da der Aufwand der Integration aber relativ hoch ist, ist eine flexible Anpassung der Lösung nicht immer so einfach möglich. Die Anzahl der bereitgestellten Dienste ist deshalb meist auf unternehmensinterne Anwendungen beschränkt und nicht universell an die Bedürfnisse der Telearbeiter anpassbar.

Die identifizierten Vor- und Nachteile der beschriebenen Lösungen im Anwendungskontext Telearbeit werden in der folgenden Übersicht (Tabelle 3-2) noch einmal zusammengefasst:

Terminaldienste	
+ direkter Zugriff auf den Büroarbeitsrechner + bekannte Arbeitsumgebung	- Bandbreite mind. DSL - Client-Installation erforderlich - aus Sicherheitsgründen nur über VPN (Einrichtungsaufwand)
Groupware	
+ gute Unterstützung der Kommunikation und Kollaboration in Teams + Abbildung schwach strukturierter Abläufe in Büroumgebungen	- geschlossene Umgebung - proprietäre Datenformate - keine Erweiterbarkeit durch weitere Dienste
Workflowmanagementsysteme	
+ Abbildung der Geschäftsprozesse + Kontrolle über den Informationsfluss + Steuerung der Kommunikation und Kooperation zwischen den beteiligten Bearbeitern	- festgelegte Abläufe, wenig Flexibilität - proprietäre Datenformate - keine Erweiterbarkeit durch weitere Dienste
Webbasierte kooperative Systeme	
+ webbasiert, Zugriff über Browser + Vielzahl von Kommunikations- und Kooperationswerkzeugen	- geschlossene Umgebung - proprietäre Datenformate - keine Erweiterbarkeit durch weitere Dienste
Enterprise Portals	
+ einheitlicher webbasierter Zugriff auf Unternehmensanwendungen + personalisierbar + an verschiedene Endgeräte adaptierbar	- meist auf ein Unternehmen beschränkt - Integration weiterer Dienste aufwendig (Portletprogrammierung)
Enterprise Application Integration	
+ Integration heterogener Anwendungen und Legacy-Systeme + prozessorientierte als auch nachrichtenorientierte Verknüpfung	- Integration der Anwendungen, aber nicht der Benutzeroberflächen - Integration weiterer Dienste aufwendig

Tabelle 3-2: Bewertung der untersuchten Lösungen

Als Fazit kann man feststellen, dass die angestrebte universelle Lösung für Telearbeitsanwendungen mit den einzelnen beschriebenen Ansätzen bisher nicht umsetzbar ist, da sie nicht den geforderten Grad an Flexibilität und Anpassungsfähigkeit bieten. Mit einer Kombination der middlewarebasierten Ansätze von EAI zur prozessorientierten Integration und von Portalen zur Vereinheitlichung der Benutzeroberflächen sollte es möglich sein, die Vorteile der vorhandenen Lösungen in einem integrierten Ansatz zu nutzen.

KAPITEL 4
BASISTECHNOLOGIEN ZUR UMSETZUNG SERVICEORIENTIERTER ARCHITEKTUREN

„Nichts auf der Welt ist so stark, wie eine Idee,
deren Zeit gekommen ist."

Victor Hugo

Das folgende Kapitel beschäftigt sich mit der Fragestellung, mit welchen Technologien serviceorientierte Architekturen (SOA) implementiert werden können und welche Vorteile der Einsatz dieser Technologie im Telearbeitskontext bringt. SOA ist ein Architektur-Ansatz, der meist im Zusammenhang mit Web Services erwähnt wird, aber prinzipiell auch mit anderen Middleware-Technologien wie z.B. CORBA (*Common Object Request Broker Architecture*), Java RMI (*Remote Method Invocation*) und DCOM (*Distributed Component Object Model*) umsetzbar ist.

Diese verschiedenen Möglichkeiten der praktischen Umsetzung sollen im folgenden Kapitel kurz vorgestellt und bewertet werden. Im zweiten Teil werden dann Web Services und die zugrunde liegenden Protokolle detaillierter spezifiziert. Neben den Basisprotokollen SOAP, WSDL (*Web Service Description Language*) und UDDI (*Universal Description, Discovery and Integration*) werden auch Protokolle und Lösungsansätze zur Komposition komplexer Web Services und für die automatisierte Suche und Auswahl von Web Services analysiert, da diese für die erfolgreiche Umsetzung einer flexiblen serviceorientierten Architektur von großer Bedeutung sind.

4.1 Architekturmodell - Serviceorientierte Architektur

4.1.1 Begriffsdefinition „Serviceorientierte Architektur"

Der Begriff „Serviceorientierte Architektur" (SOA) hat sich in den letzten Jahren zu einem viel zitierten Schlagwort entwickelt. Aber was steckt eigentlich hinter dieser Technologie? Zur Klärung der Frage, was eine serviceorientierte Architektur nun auszeichnet, wurden verschiedene Definitionen aus der Literatur analysiert.

Jeckle [Jec03] beschreibt die Methoden der SOA z.B. wie folgt:

> *„SOA ist eine Methode zur Konzeption und Realisierung von Unternehmensanwendungen, die es verschiedenen Applikationen unabhängig vom zugrunde liegenden Betriebssystem und der gewählten Programmiersprache erlaubt, Daten auszutauschen und zu verarbeiten. Zur Realisierung werden vollständige Anwendungen oder Teile daraus als Dienste angeboten, die ohne Codierungsaufwände genutzt werden können."*

Eine weitere sehr treffende Definition konnte [Cer02] entnommen werden:

> *„SOA is an architectural style whose goal is to achieve loose coupling among interacting software agents. A service is a unit of work done by a service provider to achieve desired end results for a service consumer."*

Die wichtigsten Kriterien einer dienstorientierten Architektur sind demzufolge:

- verteilte Dienste als unabgängige Komponenten,
- Beschreibung von Dienstschnittstellen,
- Nutzung von Standards, um Plattform- und Programmiersprachenunabhängigkeit zu gewährleisten,
- lose Kopplung der Dienste,
- Infrastruktur zum Veröffentlichen und Suchen von Diensten.

Im Vordergrund einer SOA stehen allein die standardisierten Schnittstellen der Anwendungen und Geschäftskomponenten; wie diese im Inneren realisiert sind, bleibt für den Anwender transparent. Durch diese Kapselung wird die Unabhängigkeit von Plattformen und Programmiersprachen ermöglicht. SOA bietet damit die Möglichkeit, sowohl einfache und zusammengesetzte Dienste als auch Legacy-Anwendungen in eine einheitliche IT-Landschaft zu integrieren (siehe Abbildung 4-1)

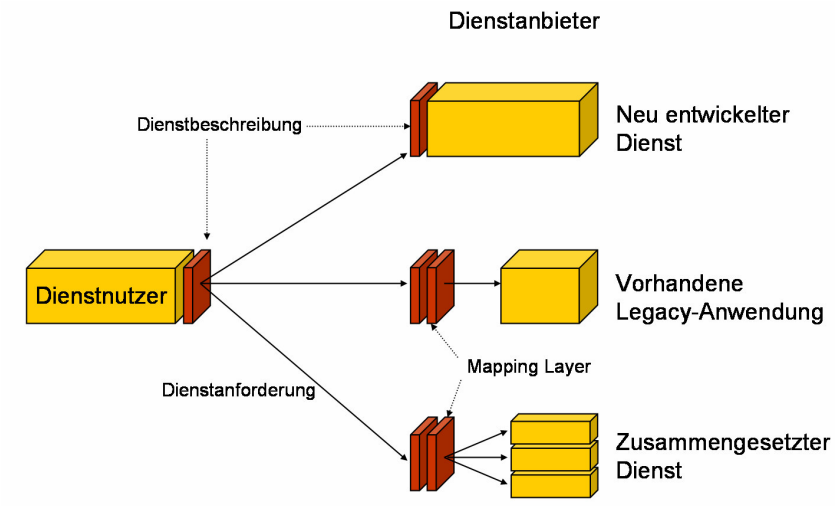

Abbildung 4-1: Verknüpfung unterschiedlichster Anwendungen in einer SOA [NeL05]

Aufgabe einer SOA ist es, Anwendungen und zugrunde liegende Implementierungen zu entkoppeln und von ihrer heterogenen Hard- und Software zu abstrahieren. Dabei steckt hinter SOA weit mehr als nur ein Technologie-Konzept. Die Umsetzung einer serviceorientierte Architektur bedarf eines ganzheitlichen Vorgehensmodells und Architekturkonzeptes bei der Abbildung der Geschäftsprozesse im Unternehmen und hat dann das Potenzial, Geschäftsziele und IT-Systeme in Einklang zu bringen.

Das grundlegende Interaktionsschema einer serviceorientierten Architektur kann man also wie folgt zusammenfassen: Anwendungen machen ihre Funktionen als Dienste über klare Schnittstellen verfügbar oder nutzen die Dienste anderer Anwendungen. Jede einzelne Applikation kann sowohl als Dienstanbieter als auch als Konsument auftreten. Basis ist eine Infrastruktur, die das Beschreiben, Suchen und Interagieren von Diensten aktiv unterstützt. Zu dieser Infrastruktur gehört neben dem Diensterbringer (*Service Provider*) und dem Dienstnutzer (*Service Consumer*) auch ein Dienstverzeichnis (*Service Broker*), welches die Schnittstellenbeschreibungen verwaltet und eine Suche nach geeigneten Diensten erlaubt. Abbildung 4-2 veranschaulicht diese drei Rollen in einer SOA und deren Beziehungen untereinander.

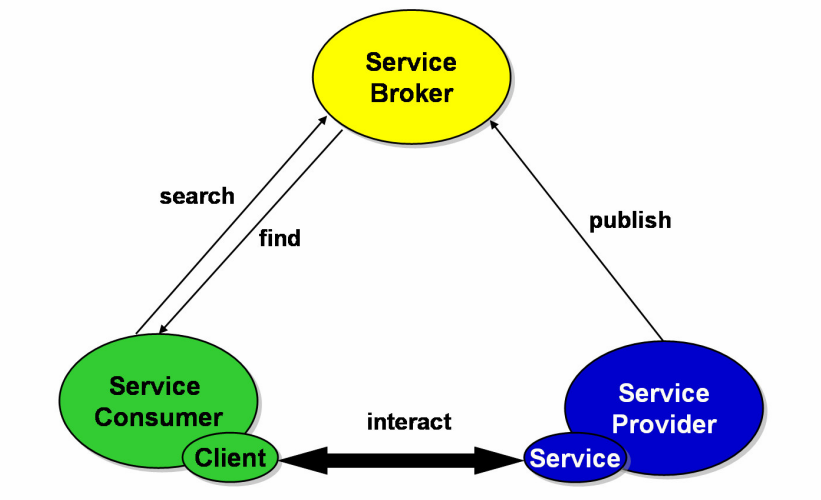

Abbildung 4-2: Prinzip einer serviceorientierten Architektur

4.1.2 Vergleich und Bewertung der Ansätze zur Umsetzung serviceorientierter Architekturen

Zur Umsetzung serviceorientierter Architekturen können verschiedene Technologien genutzt werden, auch wenn in diesem Zusammenhang meist nur Web Services erwähnt werden. Grundsätzlich sind auch herkömmliche Middleware- und EAI-Ansätze wie CORBA (*Common Object Request Broker Architecture*), DCOM (*Distributed Component Object Model*), Java RMI (*Remote Method Invocation*) und XML-RPC (*XML-based Remote Procedure Call*) für die Implementierung lose gekoppelter Service-Verbünde geeignet. Diese sollen im Folgenden kurz beschrieben und bewertet werden.

Die *Common Object Request Broker Architecture* der *Object Management Group* [OMG03] wurde Anfang der Neunziger zur Integration komplexer heterogener Systeme entwickelt, damit in verschiedenen Programmiersprachen implementierte verteilte Objekte firmenweit miteinander kooperieren können. Jedes der eigentlichen Anwendungsobjekte kommuniziert nur indirekt mit anderen Objekten. Dazu werden als Middleware-Komponenten so genannte ORBs (*Object Request Broker*) genutzt, die untereinander über das *General* oder *Internet Inter-ORB Protocol* (GIOP/IIOP) kommunizieren. CORBA setzt dabei voraus, dass die Schnittstellen der Objekte genau definiert sind und verwendet zur Beschreibung eine eigene IDL (*Interface Definition Language*). Diese ist programmiersprachenunabhängig und kann beinahe in jede Programmiersprache transformiert werden. Zusätzlich bietet CORBA noch einige *Common Services* an, die einige grundlegende Funktionen implementieren, die dem Programmierer die Arbeit erleichtern, wie den *Naming Service*, den *Transaction Service* oder den *Trader Service*. CORBA hatte das Potenzial für eine weltweit anerkannte Lösung, aber durch die Komplexität der Spezifikation entstanden immer wieder inkompatible Lösungen, die einen wirklichen Durchbruch verhinderten. Auch die Namensdienst-Lösung von CORBA wurde durch das URI-Konzept vom Markt verdrängt.

Das *Distributed Component Object Model* (DCOM) von Microsoft erlaubt es, entfernte Objekte sprachübergreifend über definierte Schnittstellen in eigene Anwendungen einzubinden und im Gegenzug selbst Objekte zur Integration bereitzustellen. Dieses Prinzip ist in der Windows-Welt sehr populär, aber leider auf sie beschränkt, weshalb es für den interoperablen Ansatz einer SOA weniger geeignet ist. Microsoft hat dieses Problem erkannt und sich maßgeblich an der Entwicklung der Web-Service-Standards beteiligt und darauf aufbauend das .NET-Framework entwickelt (näheres dazu in Kapitel 4.2.2).

Java *Remote Method Invocation* (Java RMI) wurde zur Vernetzung verteilter Java-Anwendungen entwickelt und ist daher bewusst auf die Java-Welt beschränkt. Zur Beschreibung der Schnittstellen wird keine spezielle IDL sondern Java verwendet. Die Schnittstelle wird wie gehabt über das `interface`-Konstrukt beschrieben und nur um `Remote` erweitert. Ähnlich CORBA werden die Aufrufe an das entfernte Objekt über *Stubs* und *Skeletons* weitergeleitet, die untereinander über das RMI-IIOP-Protokoll kommunizieren. Auch Java RMI besitzt einen Namensdienst - die *RMIregistry*. Java RMI stellt insgesamt eine einfache und funktionsfähige Lösung dar, verteilte Java-Anwendungen zu entwickeln, eignet sich wegen seiner Beschränkung auf Java aber nicht für eine globale Vernetzung im Sinne einer SOA.

XML *Remote Procedure Calls* (XML-RPC) wurde ursprünglich aus dem gleichen Ansatz heraus entwickelt wie SOAP. Deshalb haben beide immer noch vergleichbare Basisansätze, unterscheiden sich aber in Ihrer Komplexität und Flexibilität. XML-RPC beschränkt sich im Gegensatz zu SOAP nur auf den Aufruf entfernter Prozeduren, ist somit einfacher zu implementieren, aber dafür auch weniger flexibel. In vielen Programmiersprachen existieren Module für XML-RPC, es konnte jedoch nie eine vergleichbare Industrieunterstützung erreichen wie SOAP.

Web Services sind folglich kein komplett neuer Ansatz, sondern wurden auf Basis der vorhandenen Ansätze entwickelt. Über Web Services können heterogene Softwaresysteme miteinander gekoppelt werden. Dabei werden die Funktionen der einzelnen Anwendungen nach außen über standardisierte Schnittstellen (WSDL) gekapselt und die Dienste über den Austausch von SOAP-Nachrichten lose miteinander gekoppelt. Vorteil der Web Services gegenüber den zuvor genannten Technologien ist die klare Festlegung auf XML zur Beschreibung der Protokolle und Schnittstellen und die damit erreichte Interoperabilität sowohl was die unterstützten Programmiersprachen als auch die benutzten Plattformen betrifft.

Im Gegensatz zu den eng gekoppelten Kommunikationsarchitekturen herkömmlicher Middleware, bei denen ein großer Abstimmungsaufwand durch eine Neuausrichtung der Kommunikationsumgebung entsteht [Coy02], wurden Web Services für die Anforderungen der rasch wechselnden Internetkommunikation entwickelt. Mit der Idee eines weltweiten Verzeichnisdienstes wie UDDI ging das W3C-Entwicklungskonsortium [W3C03a] weit über die bisherigen Nameservice-Ansätze in CORBA und DCOM hinaus, die diesen Dienst nur für relativ eingegrenzte Umgebungen anbieten. Web Services repräsentieren eine industrieweite Reaktion auf die Notwendigkeit, lose gekoppelte Systeme mit einer Technologie zu verbinden, die sie an keine speziellen Programmiersprachen, Komponentenmodelle oder Plattformen bindet. Beim Design der Web-Service-Technologie wurde deshalb konsequent darauf geachtet, offene Standards und möglichst einfache und damit leicht umzusetzende Protokolle zu verwenden [W3C02a].

Vor allem für die Vermeidung von technologischer Inkompatibilität durch proprietäre Protokolle, bauen die Web-Service-Standards auf bereits etablierten Technologien wie XML und HTTP auf. XML mit ihrer erweiterbaren Syntax eignet sich hervorragend für den Datenaustausch entfernter Softwarekomponenten, auch wenn sie in unterschiedlichen Sprachen auf unterschiedlichen Plattformen implementiert sind. Hierdurch werden vielfältige Einsatzszenarien in verteilten heterogenen Systemen ermöglicht, die für Anwendungen im Internet, innerhalb von Unternehmen oder über Unternehmensgrenzen hinweg gleichermaßen geeignet sind [Knu02].

Web Services zeichnen sich vor allem durch ihre Einfachheit und Interoperabilität und die damit einhergehenden Kosteneinsparungen bei Integrationsprojekten aus. Web Services nutzen einfache, allgemein verfügbare und leicht zu handhabende Standards und sind somit

schneller und kostengünstiger zu implementieren als frühere Technologien wie CORBA. Preisgünstige und sogar kostenlose Implementierungen der Web-Service-Protokolle gibt es heute praktisch für alle aktuellen Plattformen und Programmiersprachen, wodurch Integrationsprojekte in vielen kleinen und mittleren Unternehmen erst möglich gemacht und bezahlbar werden. Mit Web Services ergibt sich zum ersten Mal eine reale Chance, die Vision serviceorientierter Architekturen Wirklichkeit werden zu lassen und das Internet als eine Sammlung von Diensten zu verstehen, die nicht nur von Menschen, sondern auch von Maschinen und anderen Anwendungen sehr effizient genutzt werden können.

Einziger, aber nicht unbedeutender Nachteil der Web-Service-Technologie, der die Nutzung in einigen Anwendungsszenarien verhindert, ist die schlechte Performanz. Da die Protokolle alle auf XML basieren, entsteht ein beträchtlicher Overhead. Es gibt eine Reihe von Untersuchungen (u.a. [EbF03]), die belegen, dass die Verarbeitung von SOAP-Nachrichten erheblich länger dauert als die von Nachrichten in anderen Middleware-Lösungen. Im Vergleich zu CORBA und RMI verdoppelt sich die Menge der übertragenen Bytes fast [EbF03]. Aus diesem Grund eignen sich Web Services nicht für den Einsatz in wirklich zeitkritischen Anwendungsszenarien. Die im Rahmen dieser Arbeit betrachteten Telearbeitsszenarien sind in dieser Hinsicht aber eher unkritisch.

4.2 Web Services

Mit der Entwicklung der Web-Service-Technologie wurde eine universelle Schnittstelle zur Kopplung von verschiedensten Anwendungen über das Internet geschaffen. Die Web-Service-Technologie bietet die optimalen Voraussetzungen, um eine SOA in der Praxis zu implementieren. Durch Web-Service-Infrastrukturen lassen sich Dienste aller Art von jedem Ort und nicht nur auf stationären Computern nutzen. Durch die "Schlankheit" der zugrunde liegenden Technologien unterstützen sie nahezu beliebige Endgeräte, einen Internet-Zugang vorausgesetzt.

Aber was sind eigentlich Web Services? In der Literatur gibt es sehr viele verschiedene Definitionen und Beschreibungen dieses Begriffes. Der Übersicht wegen sollen hier nur einige wenige Definitionen exemplarisch erwähnt werden. Die offizielle Definition des WWW-Konsortiums (W3C) lautet:

> *„A Web service is a software application identified by a URI*[39]*, whose interfaces and bindings are capable of being defined, described, and discovered as XML artifacts. A Web service supports direct interactions with other software agents using the XML-based messages exchanged via Internet-based protocols."* [W3C02a]

Weiterhin hat das W3C noch folgende Definition veröffentlicht:

> *„A Web Service is a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable format (specifically WSDL). Other systems interact with the Web Service in a manner prescribed by its description using SOAP-messages, typically conveyed using HTTP with an XML serialization in conjunction with other Web-related standards."* [W3C04]

Die folgende Definition wurde von IBM veröffentlicht, die ganz wesentlich an der Standardisierung der Web-Service-Protokolle beteiligt waren:

[39] Uniform Resource Identifier – Ein URI ist ein einheitlicher Bezeichner, der zur Identifizierung abstrakter oder physikalischer Ressourcen dient. (RFC 1630 [RFC94])

> *„Web Services are self-contained, modular applications that can be described, published, located and invoked over a network, generally, the World Wide Web."* [IBM00]

Zu guter Letzt sei hier noch die sehr allgemeine Definition von Cerami [Cer02] erwähnt:

> *„A Web Service is any piece of software that makes itself available over the Internet and uses a standardized XML messaging system."*

Betrachtet man diese und weitere Spezifikationen genauer, so erhält man die folgenden Kriterien, die Web Services genau beschreiben: Web Services sind selbst beschreibende, gekapselte Software-Komponenten. Sie bieten eine Schnittstelle an, über die ihre Funktionen entfernt aufgerufen werden können und die mit einer XML-basierten standardisierten Beschreibungssprache beschrieben werden kann. Web Services können lose durch den Austausch von XML-basierten Nachrichten miteinander gekoppelt werden. Damit ein Web Service weltweit eindeutig aufrufbar ist, wird er, wie auch herkömmliche Webseiten im Internet, mit einem URI adressiert. Um Web Services verwalten und finden zu können, wurde ein ebenfalls XML-basierter zentraler Verzeichnisdienst eingeführt. Zur Erreichung universeller Interoperabilität werden für die Kommunikation die herkömmlichen Kanäle des Internets wie HTTP oder SMTP verwendet.

4.2.1 Spezifikation und Standardisierung der Protokolle und Dienste

Web Services basieren auf den drei Basis-Standards WSDL (*Web Service Description Language*), SOAP[40] und UDDI (*Universal Description, Discovery and Integration*). Mit WSDL wird die Schnittstelle eines Web Service spezifiziert, via SOAP werden Prozeduraufrufe und Dokumente übermittelt und mit UDDI, einem zentralen Verzeichnisdienst für angebotene Web Services, können vorhandene Web Services gefunden werden [GI03].

Zusätzlich werden einige schichtenübergreifende Protokolle benötigt, die Probleme wie die Sicherheit und das Management von Web Services adressieren. Diese ergeben zusammen den *Web Service Protocol Stack* (siehe Abbildung 4-3), der von der W3C *Web Services Architecture Group* standardisiert wurde. Darauf aufbauend existieren weitere Protokolle zur semantischen Beschreibung und zur Komposition von komplexen Web Services.

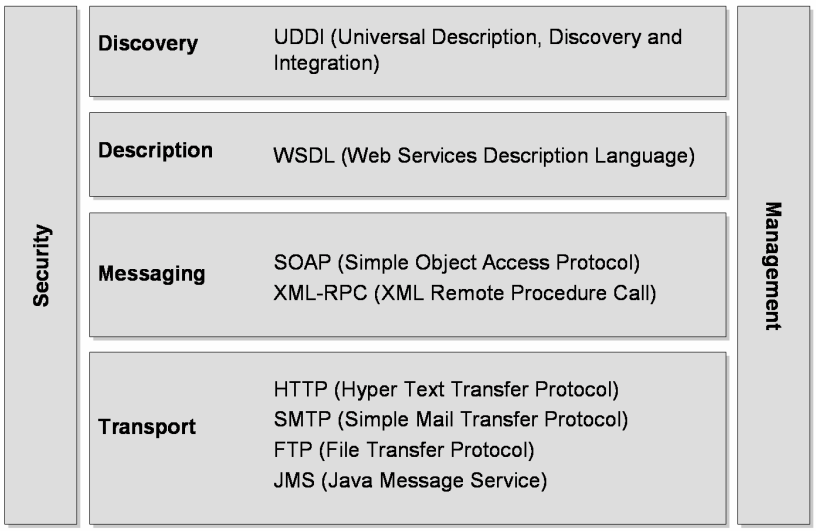

Abbildung 4-3: Web Services Protocol Stack

[40] Ab Version 1.2 (Juni 2004) wird „SOAP" nur noch als Eigenname benutzt. Die bisherige Verwendung als Akronym für „Simple Object Access Protocol" entfällt damit.

In den folgenden Kapiteln werden die einzelnen Protokolle noch näher spezifiziert. Auf Grund der Mitarbeit vieler verschiedener Unternehmen und Institutionen am Standardisierungsprozess im Bereich Web Services und der damit verbundenen permanenten Neuvorstellung von Protokollen und Spezifikationen, kann hier kein Anspruch auf Vollständigkeit erhoben werden und deshalb werden an dieser Stelle nur die wichtigsten Basisprotokolle beschrieben. Sollten weitere Protokolle bei der Implementierung relevant sein, werden sie an entsprechender Stelle vorgestellt und näher spezifiziert.

4.2.1.1 SOAP

Der Datenaustausch zwischen Web-Service-Anwendungen wird durch SOAP [W3C00] realisiert. SOAP wurde 1999 ursprünglich von Microsoft entwickelt und 2001 vom W3C standardisiert. SOAP ist ein zustandsloses *one-way* Nachrichtenaustauschformat, welches auf XML basierend einen Austausch von strukturierten und vordefinierten Informationen in einer verteilten Umgebung ermöglicht. Durch die Nutzung von XML ist SOAP ein plattform- und programmiersprachenunabhängiges Protokoll, welches auf TCP/IP oder höheren Kommunikationsprotokollen wie HTTP oder SMTP aufsetzt.

Dieses standardisierte XML-basierte Protokoll stellt den zu übertragenden Informationen einen *Envelope* (Umschlag) bereit, mit dem die Anwendungsdaten verpackt werden können [STK02]. Für den eigentlichen Transport der Daten zwischen den Anwendungen muss SOAP jedoch auf ein Transportprotokoll zurückgreifen. Dieser Umstand ist beabsichtigt und soll sowohl zur Einfachheit des SOAP-Protokolls beitragen [W3C00] als auch den Einsatz jener Transportprotokolle ermöglichen, die sich bei der Kopplung von verteilten Internetanwendungen bereits bewährt haben [Coy02]. Der Aufbau von SOAP wird durch das SOAP *Processing Model* im Standard erläutert. Im Folgenden soll kurz auf die grundlegenden Merkmale eingegangen werden. Eine SOAP-Nachricht besteht aus einem *Header* (Kopf) und einem *Body* (Körper), welche in einem *Envelope* (Umschlag) verpackt sind, so dass man den Inhalt einfacher in einem verteilten System versenden kann. Die Struktur einer SOAP-Nachricht ist in Abbildung 4-4 dargestellt.

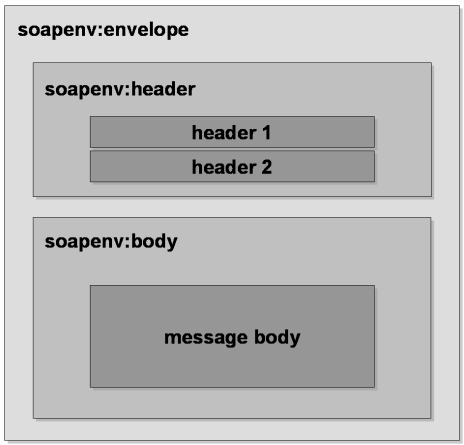

Abbildung 4-4: Struktur einer SOAP-Nachricht

Der SOAP-*Header* kann vom Empfänger und so genannten „Intermediären“, Netzknoten, die die SOAP-Nachricht weiterleiten und auch bearbeiten können, gelesen und ausgewertet werden. Er bietet bei einem Nachrichtenaustausch die Möglichkeit, einige Zusatzinformationen, wie zum Beispiel eine Session-ID oder einen Authentisierungsschlüssel, zu übertragen. Jede Station zwischen Sender und Empfänger ist ein *Intermediary* (Vermittler). Diese werden ebenfalls durch einen *Uniform Resource Identifier* (URI) identifiziert. Das Attribut *role* legt dessen Vorgehensweise bei der Behandlung der SOAP-Nachricht fest.

Der SOAP-*Body* enthält die eigentlichen Nutzdaten und daneben auch einen Bereich für *fault messages* (Fehlermeldungen), falls es zu Übertragungsfehlern kommt. Er besitzt im Gegensatz zum *Header* keine Attribute, mit denen man den Nachrichtenfluss beeinflussen kann. Der *Body* wird automatisch beim Empfänger ausgeführt. Falls dabei ein Fehler auftreten sollte, wird eine vordefinierte Fehlermeldung erzeugt und zurück gesendet.

Nachdem der Aufbau einer SOAP-Nachricht dargelegt wurde, soll an dieser Stelle kurz auf das eigentliche Protokoll eingegangen werden. Das SOAP-Protokoll kennt *Sender* (Sender), *Receiver* (Empfänger) und *Intermediary* (Vermittler). Eine SOAP-Nachricht wird von einem Sender unter zu Hilfenahme des *Header* an einen Empfänger gesendet. Auf dem Weg dorthin wird der Nachrichtenkopf von Intermediären ausgewertet. Wie bereits beschrieben, kann als Host-Transportprotokoll prinzipiell jedes beliebige andere Protokoll verwendet werden, das genügend große Nutzdaten transportieren kann. Wie SOAP in den OSI-Protokoll-Stack eingeordnet werden kann, ist in Abbildung 4-5 zu sehen. Die Zusammenarbeit von SOAP mit dem Host-Protokoll wird durch das SOAP *Protocol Binding Framework* definiert.

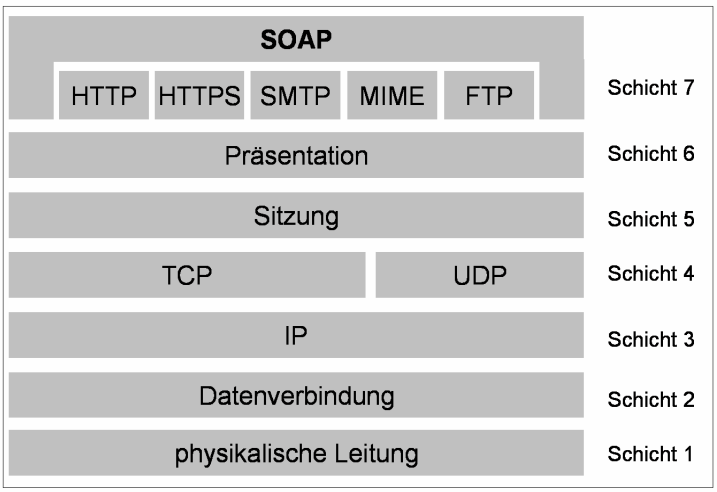

Abbildung 4-5: Einordnung von SOAP in das OSI-Schichtenmodell

Um den Nachrichtenaustausch zu unterstützen bietet SOAP das SOAP *Extensibilitiy Model* an. Dieses liefert dem Anwender schon definierte *Message Exchange Patterns* (Muster zum Nachrichtenaustausch). Es ist möglich, einen Nachrichtenaustausch dialogorientiert mithilfe des *Response Message Exchange Pattern* oder RPC-orientiert mit dem *Request Response Message Exchange Pattern* zu gestalten. Bei einem RPC-orientierten Nachrichtenaustausch bietet sich natürlich zum Transport das HTTP-Protokoll an, welches auch nach dem Muster *Request-Response* funktioniert. Natürlich kann man auch, wie in Abbildung 4-5 zu sehen ist, andere Transportprotokolle nutzen. In der Praxis haben sich allerdings die standardisierten Protokolle HTTP (*Hypertext Transfer Protocol*), HTTPS (*HTTP Secure*) und SMTP (*Simple Mail Transfer Protocol*) für den Transport von SOAP-Nachrichten durchgesetzt.

Als Rückgabewert erhält man in jedem Fall eine SOAP-*Response*-Nachricht, deren Quellcode in Listing 4-1 zu sehen ist. Der Rückgabewert ist im Element `result` enthalten.

```
SOAP 1.2
<soapenv:Body>
  <getResponse soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
    <rpc:result>95224;95228564;123345;</rpc:result>
  </getResponse>
</soapenv:Body>
```

Listing 4-1: SOAP Response Nachricht

Die hier beschriebenen Elemente und Attribute sowie die Eigenschaften von SOAP sind die Grundlagen um einen programmiersprachenunabhängigen Datenaustausch zwischen heterogenen verteilten Systemen zu ermöglichen.

4.2.1.2 WSDL - Web Services Description Language

Um Web Services und ihre Schnittstellen allgemeingültig und programmiersprachenunabhängig zu beschreiben, wird die XML-basierte *Web Services Description Language* (WSDL) genutzt. Die Aufgabe der WSDL besteht darin, eine Schnittstelle zwischen einem bereitgestellten Dienst und einem anfragenden Client zu spezifizieren. Ist einem Client die WSDL-Beschreibung eines Dienstes bekannt, kann er den Web Service lokalisieren und jede der öffentlich zugänglichen Operationen des Dienstes aufrufen.

WSDL ist eine Weiterentwicklung und Zusammenführung der Standards *SOAP Contract Language* (SCL) von Microsoft, *Service Description Language* (SDL) und *Network Accessible Service Specification Language* (NASSL) von IBM. Bereits im Jahr 2000 wurde beim W3C die Version 1.0 als W3C Note eingebracht. Maßgeblich verantwortlich für deren Entwicklung sind Ariba, IBM und Microsoft. Die Version 2.0 befindet sich seit Oktober 2004 im *Last Call Status* und soll Anfang 2006 veröffentlicht werden [W3C05].

WSDL besitzt eine Vielzahl von Gemeinsamkeiten mit den *Interface Definition Languages* (IDL) von CORBA oder Microsoft. Diese Beschreibungssprachen haben zum Ziel, Schnittstellen in Form von Methodensignaturen sowie zugehörige Datentypen zu spezifizieren. Jedoch bietet WSDL ein weitaus größeres Maß an Erweiterbarkeit als dies bei IDLs der Fall ist. So ist es z.B. möglich, Kommunikationsendpunkte für Nachrichten zu definieren, ohne das Format der Nachricht oder das Protokoll für den Datenaustausch zu kennen [HeZ03]. Die Verwendung der WSDL-Beschreibung ermöglicht es, Web Services automatisch zu propagieren und neue Funktionalität mit einem minimalen Anpassungsaufwand an den Server- und Client-Anwendungen zu entwickeln.

Web Services werden in WSDL mit Hilfe der folgenden sechs Hauptelemente beschrieben, deren Abhängigkeiten in Abbildung 4-6 dargestellt sind:

`Types`: beschreibt alle Datentypen, welche zwischen Client und Server ausgetauscht werden.

`Messages`: repräsentiert eine abstrakte Beschreibung der auszutauschenden Nachrichten. Ein solches Element besteht aus dem Namen der Nachricht und mehreren optionalen logischen Teilen (`parts`), die Anfrageparameter bzw. Rückgabewerte festlegen.

`Port Types`: beschreibt eine Menge von abstrakten Operationen. Jede dieser Operationen verweist dabei auf die Definition einer Eingangsnachricht und mehrerer Ausgangsnachrichten (`messages`).

`Binding`: definiert ein konkretes Protokoll und Datenformat-Spezifikationen für Operationen und Nachrichten eines einzelnen `Port Type`.

`Port`: spezifiziert die Adresse für ein Binding und somit einen einzelnen Kommunikations-Endpunkt.

`Service`: setzt eine Menge zusammengehöriger Ports in Relation zueinander.

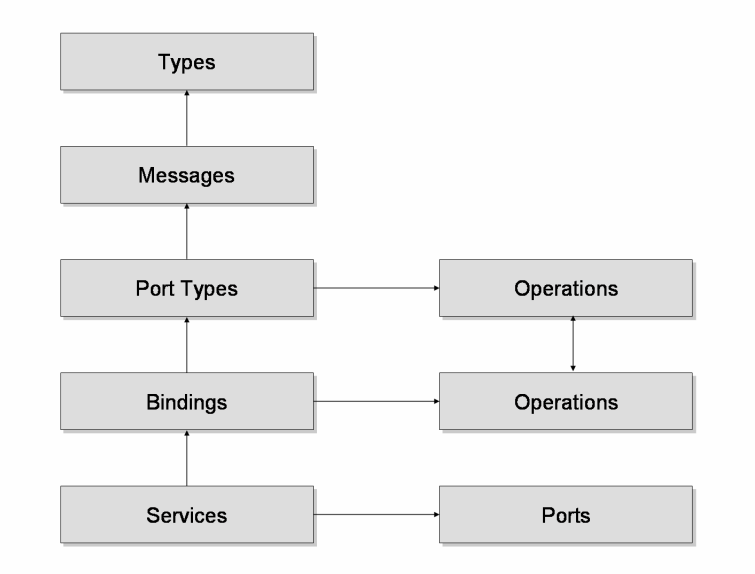

Abbildung 4-6: Aufbau eines WSDL-Dokumentes

Eine WSDL-Datei besteht dabei aus zwei Teilen - einem abstrakten und einem konkreten Teil. Im abstrakten Teil werden unabhängig von einem konkreten Protokoll oder Service die Operationen und verwendeten Datentypen beschrieben. Der abstrakte Teil enthält die Elemente `types`, `message` und `portType` und beschreibt neben dem Namen des Web Service auch die Parameter zum Aufruf des Dienstes und die zu erwartenden Antworten. Der zweite Teil der WSDL enthält die konkreten Informationen, die zum Aufruf des Dienstes benötigt werden. Dort wird beschrieben, über welche URI (*Uniform Resource Identifier*) und Protokolle ein Web Service erreicht werden kann und wie die Daten serialisiert und codiert werden. Das Element `binding` beschreibt für jeden `portType` ein konkretes Protokoll und Format zum Nachrichtenaustausch. Listing 4-2 zeigt einen Ausschnitt aus der WSDL des Amazon Web Service [Ama05].

```
<?xml version="1.0" encoding="UTF-8" ?>
<definitions …>

  <types>
    <schema>…</schema>
  </types>
  <messages>
    <part>…</part>
  </messages>
  <portType>
    <operation>
      <input>…</input>
      <output>…</output>
    </operation>
  </portType>
```

Abstrakter Teil

```
  <binding>
    <operation>
      <input>…</input>
      <output>…</output>
    </operation>
  </binding>
  <service name="AWSECommerceService">
    <port name= "AWSECommerceServicePort" binding="tns:AWSECommerceServiceBinding">
      <soap:address location="http://soap.amazon.de/onca/soap?
       Service=AWSECommerceService"/>
    </port>
  </service>

</definitions>
```

Konkreter Teil

Listing 4-2: Auszug aus der WSDL des Amazon Web Service

4.2.1.3 UDDI - Universal Description, Discovery and Integration

Wie bereits erwähnt, ist es bei der Umsetzung einer SOA von hoher Wichtigkeit, einen Ort zur Verfügung zu stellen, an dem verschiedene Informationen über verfügbare Dienste (Schnittstellen, Beschreibung, Protokoll, Lokalisation etc.) zentral und durchsuchbar verwaltet werden können.

Mit dem Ziel einer Standardisierung solcher Verzeichnisdienste und der Schaffung eines offenen und plattformunabhängigen Frameworks haben sich Unternehmen wie SAP, IBM, Microsoft, SUN und andere zu der Initiative UDDI.org [UDD05] zusammengeschlossen. Die UDDI-Spezifikation 1.0 wurde im September 2000 von Microsoft, IBM und Ariba veröffentlicht. Mit der Version 3.0 von UDDI wurde die Standardisierung an OASIS übergeben, welche UDDI 3.0 im Februar 2005 standardisiert hat [OAS05a]. Ein unabhängiges Konsortium aus den genannten Unternehmen verwaltet frei zugängliche UDDI-Register, die ihre öffentlichen Daten untereinander abgleichen. Dadurch wird ein globales Register UBR (*UDDI Business Registry*) publiziert, welches es in erster Linie allen Unternehmen weltweit ermöglichen soll, ihre angebotenen Dienste zugänglich zu machen, sich gegenseitig zu finden und so zu kooperieren.

Die Aufgabe von UDDI ist es, eine standardisierte Methode für das Veröffentlichen und Auffinden von Web Services zur Verfügung zu stellen. In diesem Register können angebotene Services publiziert, Schnittstellen, Protokolle etc. beschrieben und bereichsspezifische Taxonomien für Web Services etabliert werden. Während die Informationen im WWW unstrukturiert und abstrakt abgelegt sind, da es kein verbindliches, einheitliches Format gibt, ist innerhalb eines UDDI-Registers das Format für jeden Zweck entsprechend vorgegeben, was die Erstellung von Suchanfragen sehr vereinfacht. Bei der Entwicklung von UDDI wurden, analog zu bekannten Registern im Telekommunikationsbereich, folgende drei Hauptkategorien eingeführt, um gezielt nach Web Services und deren Anbietern suchen zu können: *White Pages*, *Yellow Pages* und *Green Pages*. Dabei beinhalten die *White Pages* lediglich Kontaktinformationen zum Web-Service-Anbieter, die *Yellow Pages* fassen die Anbieter nach Branchen und Leistungen zusammen, während die *Green Pages* die technischen Informationen zu den veröffentlichten Diensten bereitstellen. Diese Informationen werden mit Hilfe von XML-Schemata erfasst, verwaltet und verteilt. Dadurch ist eine Automatisierung der Suche und der Auswahl geeigneter Web Services möglich.

Ein UDDI-Register besteht dabei aus den vier Datentypen *business information*, *service information*, *binding information* und *specifications of services*, welche in diversen Entities im Register festgehalten werden. Die wichtigsten Entities werden im Folgenden kurz erläutert:

- `businessEntity`: speichert die grundlegenden Geschäftsinformationen des Web-Service-Anbieters. Diese bestehen aus dem Firmennamen und der Adresse sowie weiteren Kontaktinformationen und einer Kategorisierung. Sie ist über einen `identifier` immer eindeutig identifizierbar und kann mehrere `businessServices` enthalten.
- `businessService`: beschreibt einen Web Services und gehört genau zu einer `businessEntity`. Es verweist meistens nur auf eine Art von Web Service, enthält aber für diesen unterschiedliche Adressen. So können unter diesen Adressen unterschiedliche Versionen des Dienstes laufen, die sich z.B. durch verschiedene Verbindungsprotokolle unterscheiden.
- `bindingTemplate`: beinhaltet die technischen Aspekte eines Web Service und gehört genau zu einem `businessService`. Das `bindingTemplate` definiert über das `accessPoint`-Element die Adresse, unter der der Web Service zu finden ist und liefert auch die Verknüpfungen zu den dazugehörigen `tModels`.

- `tModel`: steht für *Technical Model* und ist ein Container für jede Art von Spezifikation. So könnte es zum Beispiel eine WSDL-Schnittstelle, eine Klassifikation, eine Interaktionsanweisung oder eine menschenlesbare Beschreibung eines Dienstes beherbergen. Die Informationen, die dort abgelegt sind, dienen zum Vergleich, ob ein Dienst tatsächlich die gewünschten Anforderungen erfüllt. Ein Dienst kann mehrere `tModels` haben, die verschiedene Ausprägungen z.B. der Funktionalität abbilden.
- `publisherAssertion`: wird benutzt, um eine Beziehung zwischen zwei `businessEntity`-Strukturen zu beschreiben. So kann ein Konzern z.B. auf seine Konzerntöchter und diese wiederum auf ihre Niederlassungen referenzieren. Auch die Definition von Partnerschaften zwischen zwei gleichberechtigten Unternehmen oder von Lieferanten-Abnehmer-Relationen ist möglich. Eine Beziehung zweier Unternehmen ist nur dann öffentlich sichtbar, wenn beide Partner eine äquivalente Beschreibung der `publisherAssertion` veröffentlicht haben.

Die Zusammenhänge der beschriebenen elementaren UDDI-Datenobjekte sind in Abbildung 4-7 veranschaulicht.

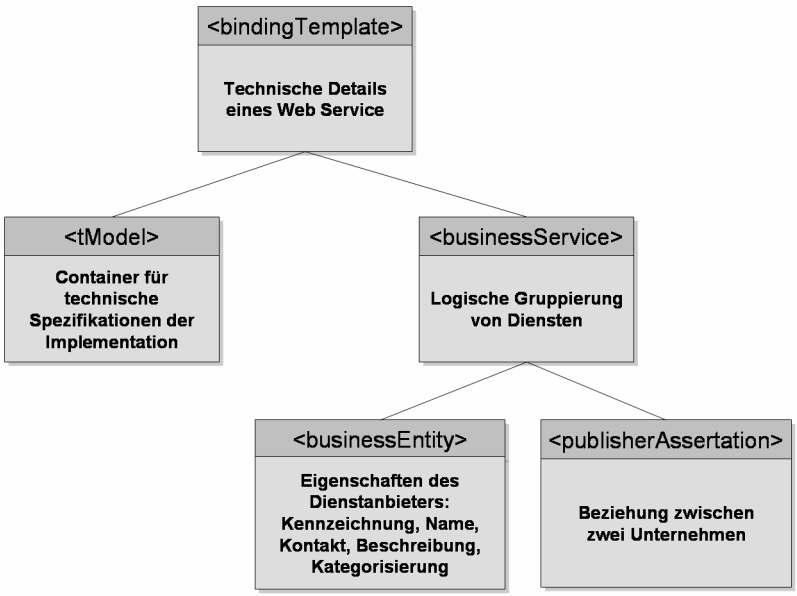

Abbildung 4-7: Elementare UDDI-Datenstrukturen (in Anlehnung an [HeZ03])

Um einen Web Service zu veröffentlichen, muss man nun lediglich die Elemente `businessEntity`, `businessService`, `bindingTemplate` und die notwendigen `tModel` zum Web Service passend erstellen und im UDDI-Verzeichnis veröffentlichen.

Für die Generierung von Anfragen an eine UDDI-Registry, um z.B. nach einem Unternehmen oder angebotenen Diensten zu suchen, existieren eine ganze Reihe von Nachrichtentypen, die man grob in die folgenden fünf Kategorien einordnen kann:

- **Authentifikation**: Diese Nachrichten dienen dazu, den Nutzer bei der Anmeldung zu authentifizieren.
- **Suchen**: Diese Kategorie umfasst alle Nachrichten, die zum Auffinden von Diensten dienen und deren SOAP-*Body* mit dem Element `find` beginnt.
- **Detailinformationen abfragen**: Mit diesem Nachrichtentyp kann man Detailinformationen der Entities abfragen.

- **Hinzufügen, Ändern**: Die Nachrichtentypen zum Hinzufügen und Ändern sind gleich aufgebaut und die Registry erkennt automatisch, ob ein Eintrag schon vorhanden ist und geändert wird, oder ob er neu angelegt werden muss. Sie setzen eine vorangegangene Authentifizierung voraus.
- **Löschen**: Diese Nachrichten dienen zum Löschen von UDDI-Einträgen und setzen ebenfalls eine Authentifizierung voraus.

Alle hierbei ausgetauschten Nachrichten sind SOAP-Nachrichten. Die UDDI-Spezifikation umfasst keine direkte Definition einer Programmierschnittstelle, um diese Suchanfragen zu implementieren, sondern empfiehlt generell die Nutzung von *Remote Procedure Calls* via SOAP. Fast alle UDDI-Registry-Anbieter stellen eine webbasierte Oberfläche zum manuellen Suchen oder APIs zum automatisierten Suchen aus Programmen heraus zur Verfügung.

4.2.1.4 Zusammenfassung

Mit den vorgestellten Basisprotokollen SOAP, WSDL und UDDI stehen der Web-Service-Technologie alle wesentlichen Werkzeuge zur Unterstützung einer serviceorientierten Architektur zur Verfügung. Wie die soeben beschriebenen Protokolle im Anwendungsfall zusammenspielen, ist in Abbildung 4-8 dargestellt.

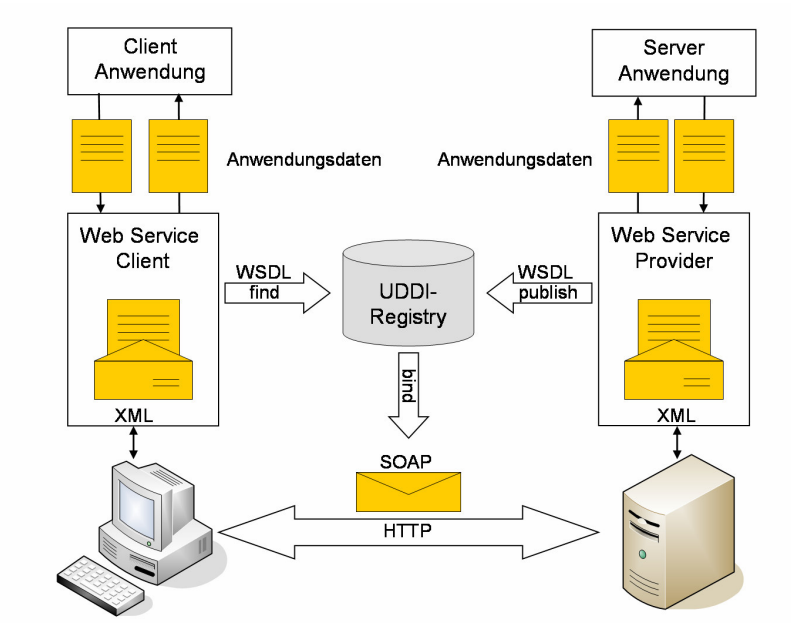

Abbildung 4-8: Zusammenspiel der Web-Service-Protokolle

Eine Client-Anwendung stellt eine Anfrage an den *Service Broker*, also ein entsprechendes UDDI-Register, und erhält die WSDL-Schnittstelle des vom *Service Provider* angebotenen Web Service. Jetzt kann zwischen Client und Server eine HTTP-Verbindung aufgebaut werden, über die beide Parteien mittels SOAP-Nachrichten kommunizieren. Die Server-Applikation verarbeitet die Anfrage und sendet die XML-codierten Ergebnisdaten an den Client zurück. Danach wird die Verbindung wieder abgebrochen und die lose Kopplung damit aufgelöst.

4.2.2 Untersuchung vorhandener Plattformen zur Implementierung von Web Services

Viele Unternehmen stehen bei der Wahl einer Middleware-Plattform zur Unterstützung von Web Services vor der Alternative Java oder .NET. Dabei haben beide Anwendungsplattformen viele Gemeinsamkeiten. Beide bieten objektorientierte Programmierung, umfangreiche Klassenbibliotheken, Komponentenmodelle und Frameworks, Funktionen zur dezentralen Verarbeitung, eine verwaltete Ausführungsumgebung sowie Funktionen zur Verarbeitung und Bereitstellung von Web Services [TMF04]. Sie unterscheiden sich lediglich durch Plattform- und Programmiersprachen-Abhängigkeit und die Unterstützung durch verschiedene Werkzeuge und Lösungsanbieter. Beide Plattformen sollen im Folgenden kurz vorgestellt werden. Ein detaillierter Vergleich ist dem Rahmen dieser Arbeit hingegen nicht angemessen, dazu sei auf [EbF03] verwiesen.

4.2.2.1 Java 2 Enterprise Edition (J2EE)

Die *Java 2 Enterprise Edition* ist eine Teilspezifikation von Java für eine ganze Palette von Middleware-Diensten. Mit dieser Spezifikation wird ein allgemeiner Rahmen für die Entwicklung verteilter Multi-Tier-Anwendungen mit modularen Komponenten zur Verfügung gestellt. J2EE bietet Werkzeuge, Dienste und Entwurfsstrukturen für die Entwicklung solcher Anwendungen an. Interoperabilität zwischen den verschiedenen Komponenten und Schichten wird über klar definierte Schnittstellen gewährleistet. Diese Architektur bringt somit die idealen Vorraussetzungen für die Umsetzung von Web Services mit.

Java-Applikationsserver werden in der Spezifikation Java 2 Enterprise Edition (J2EE) zusammengefasst und sind durch sie standardisiert. Die im April 2004 freigegebene Spezifikation J2EE 1.4 wurde um folgende Technologien zur Unterstützung von Web Services erweitert:

- *Web Services for J2EE* (WSEE),
- *Java API for XML-based RPC* (JAX-RPC),
- *SOAP with Attachments API for Java* (SAAJ),
- *Java API for XML Registries* (JAXR),
- *Java Architecture for XML Binding* (JAXB).

Ältere Versionen können aber durch die Installation des *Java Web Service Developer Pack* (WSDP) um Web-Service-Funktionalitäten erweitert werden. Viele große und etablierte Firmen wie IBM oder Borland bieten J2EE-Produkte an, andere sind erst im J2EE-Umfeld bekannt geworden wie BEA. Außerdem existiert im Java-Umfeld auch eine große Open-Source-Gemeinde, die kostenlose Werkzeuge und Server-Implementationen zur Verfügung stellt. Die Technologien für Applikationsserver sind inzwischen so ausgereift, dass zwischen den einzelnen Produkten nur noch geringe Unterschiede bestehen. Jeder Hersteller offeriert seine Plattform in verschiedenen Ausgaben - von schlanken Varianten bis zu umfangreichen Enterprise-Editionen - sowie zusätzliche Werkzeuge für die Entwicklung und Integration von Anwendungen.

IBM mit *Websphere* und Bea mit *Weblogic* halten aufgrund vieler Kunden und umfangreicher Middleware-Produkt-Paletten den Löwenanteil am Web-Services-Markt [TMF04]. Oracle und Sun liegen mit deutlichem Abstand dahinter. Durch den Einsatz von Grid-Technologien erreicht der Oracle *10g Application Server* zusätzlich eine besonders hohe Skalierbarkeit. Wenn Kosten gespart werden müssen, kommen Open-Source-Produkte wie *JBoss AS, Apache Tomcat* und *Axis* in Betracht.

4.2.2.2 Microsoft .NET-Framework

Wenn die Unternehmens-IT vorrangig durch Microsoft-Produkte geprägt ist, bietet sich Microsoft .NET als Alternative an. Microsofts .NET-Software bietet der J2EE-Plattform vergleichbare Leistungen, ist jedoch auf Windows-Umgebungen beschränkt. Das .NET-Framework verwendet Web Services als integralen Bestandteil seiner verteilten Infrastruktur. In Windows-Systemen werden Web Services wie alle anderen Programme in der *Common Language Runtime* (CLR) ausgeführt. Sie führt programmiersprachenunabhängigen Code aus, der durch den Compiler der jeweiligen Sprache in eine Zwischendarstellung in der *Intermediate Language* (IL), ähnlich dem Java Bytecode, übersetzt wird. Deshalb können Web Services in allen von .NET unterstützten Sprachen programmiert und dabei die bekannten Entwicklungswerkzeuge genutzt werden. Die IL-Fragmente werden in ausführbare Pakete, so genannte *Assemblies,* verpackt und von einem *Just in Time Compiler* (JIT) in den Maschinen-Code für das jeweilige Betriebssystem übersetzt. In der Klassenbibliothek von .NET existiert eine eigens für Web Services zuständige Bibliothek namens `System.Web`.

Das *.NET Component Model* gewährt dem Entwickler Zugriff auf die Funktionalität des Betriebssystems. Der Applikationsserver für Anwendungsprogramme ist in das Betriebssystem Windows Server 2003 integriert und besteht im Wesentlichen aus *Internet Information Services* (IIS) sowie *Active Server Pages for .NET* (ASP.NET). Mit *Visual Studio .NET* stellt Microsoft außerdem eine umfangreiche Entwicklungsumgebung für Web Services und zugehörige Clientanwendungen bereit. Das *.NET Framework SDK* generiert für eine Client-Anwendung aus der WSDL eines Web Service automatisch entsprechende Proxy-Klassen, die es ermöglichen, auf die entfernten Serverkomponenten zuzugreifen, als wären sie im eigenen Adressraum.

4.2.2.3 Fazit

Java-Applikationen sind grundsätzlich plattformunabhängig, aber im Gegenzug natürlich an die Programmiersprache Java gebunden. Bei der Entscheidung für Java bleiben Wahlfreiheiten zwischen vielen verschiedenen Softwareherstellern und Open-Source-Produkten. Microsoft hingegen unterstützt mit .NET zwar mehr als 25 Programmiersprachen, aber nur die eigenen Windows-Betriebssysteme. Weitere Unterschiede sind auch in der Standardisierung der Frameworks zu finden. Während J2EE vollständig standardisiert wurde, legte Microsoft lediglich die Teilspezifikationen CIL und C# des .NET-Frameworks einem offiziellen Gremium vor.

Ein wesentlicher Vorteil der .NET-Technologie ist die native Unterstützung bei der Entwicklung von Web Services und die Integration in das Betriebssystem. J2EE wurde dagegen lediglich um Web-Service-Technologien erweitert und nicht direkt für diesen Technologieansatz entworfen. J2EE bietet dagegen mit seinem Konzept des Applikationsservers eine Reihe von Erleichterungen bei der Programmierung und ermöglicht damit die Konzentration der Entwickler auf die Geschäftslogik der Anwendung.

Eine Entscheidung für eine der beiden Plattformen kann nur schwierig getroffen werden und hängt von einer Reihe weiterer Faktoren ab. Viele Unternehmen bevorzugen bei Middleware-Komponenten Java, weil die zugrunde liegende Java 2 Enterprise Edition standardisiert und dadurch die Interoperabilität zwischen verschiedenen Plattformen und heterogenen Anwendungen gewährleistet wird. Für desktopzentrierte Anwendungen mit umfangreicher Benutzerschnittstelle wird hingegen oft .NET favorisiert. Im Rahmen dieser Arbeit wurden Web Services genutzt, die sowohl auf der Basis von .NET als auch mit Java implementiert wurden, um die Interoperabilität des Ansatzes aufzuzeigen.

4.2.3 Komposition komplexer Dienste

Die von Web Services angebotene formale Beschreibung (WSDL) enthält Nachrichten- und Typinformationen für eine einfache bidirektionale Kommunikation, d.h. für eine Anfrage und das dazugehörige Ergebnis. Um komplexere Dienste und deren Zusammensetzung aus einfachen Diensten zu beschreiben, wird eine weitere Meta-Sprache benötigt, die den Ablauf und die Abhängigkeiten der einzelnen Services beschreibt und eine Schnittstelle des komplexen Dienstes nach außen definiert.

Mit den heutigen Kompositionssprachen ist es möglich, zusammengesetzte Web Services zu definieren, die mehrere bestehende einfache Web Services zu einem wiederverwendbaren Geschäftsprozess höherer Ordnung koppeln. Mit entsprechenden Werkzeugen können Web Services aggregiert, integriert und in jeder gewünschten Komplexität verschachtelt werden. Eine sequentielle, bedingte, parallele oder zeitbezogene Ausführung sowie andere Abhängigkeiten können in komplexen Web Services definiert werden. Dadurch kann man dem Nutzer einen komplexen Dienst anbieten, ohne dass dieser die einzelnen Web Services kennen muss, die an der Verarbeitung der Anforderung beteiligt sind.

4.2.3.1 Grundprinzipien der Web-Service-Komposition

Die Web-Service-Kompositionssprachen kann man nach der Art der Ablaufkontrolle in die Kategorien Orchestrations- und Choreographiesprachen einteilen, die im Folgenden näher erläutert werden sollen. Grundsätzlich ist aber auch eine Kombination oder Verschachtelung beider Gestaltungsprinzipien denkbar, da komplexe Dienste wiederum mit anderen Diensten interagieren können. Weiterhin lassen sich die betrachteten Kopplungsmechanismen anhand des Kompositionszeitpunktes unterteilen.

4.2.3.1.1 WS-Orchestration

WS-Orchestration verfolgt einen prozessorientierten Ansatz der Service-Komposition. Hierbei werden primitive Dienste mit Hilfe einer Kompositionssprache angeordnet und deren Ablauf genau festgelegt. Somit wird im Voraus geplant, wann und unter welchen Bedingungen ein Web Service aufgerufen wird. Es gibt folglich eine kontrollierende Instanz, die den Ablauf der Aktivitäten, die zur Zielerreichung notwendig sind, überwacht [PEL03]. WS-Orchestration baut dabei auf den Prinzipien des Workflow-Managements auf, um komplexe Geschäftsprozesse abbilden zu können. Abbildung 4-9 zeigt das Zusammenspiel der Web Services und des Koordinators.

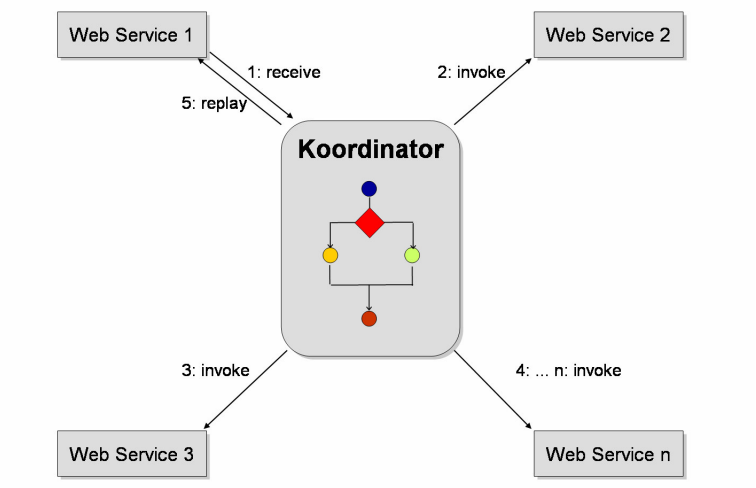

Abbildung 4-9: Prinzip der WS-Orchestration

4.2.3.1.2 WS-Choreographie

Der *WS-Choreographie*-Ansatz hingegen kommt aus dem Bereich der *Service Coordination Protocols* und hat als Grundidee die verteilte Aufgabenbearbeitung. Dabei gibt es keine Instanz, welche alles kontrolliert, sondern jede angesprochene Instanz ist für den weiteren Verlauf der Bearbeitung zur Zielerreichung verantwortlich. Der Ablauf wird über den Nachrichtenaustausch zwischen den beteiligten Web Services geregelt (siehe Abbildung 4-10).

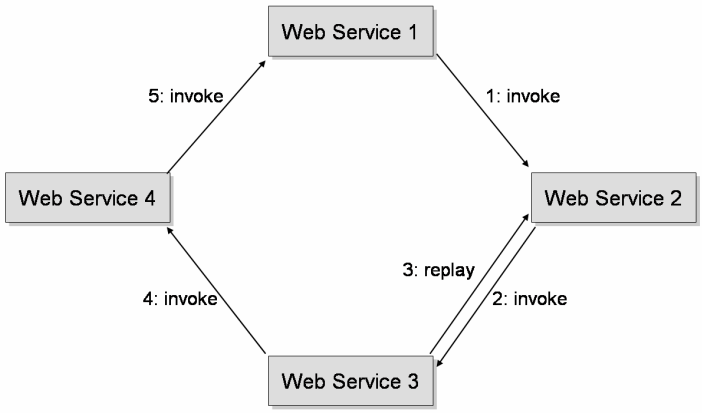

Abbildung 4-10: Funktionsprinzip von WS-Choreographie

Hierbei dominieren als Entwurfsmethodik die aus der Agententechnologie bekannten Entwurfs- und Kommunikationsmuster. Die Aufgabe der WS-Choreographie ist es, den Ablauf zwischen Client und Web Service zu regeln. Dabei spielt es jedoch keine Rolle, ob der Web Service zur Zielerreichung selbst als Client in Erscheinung tritt oder nicht. Da komplexe Web Services auch nur aus einzelnen primitiven Diensten bestehen, ist es folglich auch möglich mit Hilfe des *Service Coordination Protocols* einen komplexen Web Service zu erstellen.

4.2.3.1.3 Kompositionszeitpunkt

Es besteht in beiden zuvor beschriebenen Konzepten die Möglichkeit, dass die Komposition des komplexen Web Services schon während der Design-Zeit des Systems oder erst zur Laufzeit stattfindet. Im ersten Fall spricht man von einer statischen Komposition; hier wird schon während der Systementwicklung festgelegt, wie der Ablauf des komplexen Dienstes aussieht und unter welchen Bedingungen dieser ausgeführt wird. Dies ist momentan die gängige Methode, um komplexe Web Services zu erstellen.

Bei der dynamischen Komposition hingegen wird bei der Systementwicklung lediglich die Semantik des Geschäftsprozesses definiert, welche dann zur Laufzeit von einer oder mehreren intelligenten Instanzen interpretiert wird. Dabei müssen zur Laufzeit die Funktionen *Retrieval* (Auffinden), *Invocation* (Ansprechen) und *Combination* (Verknüpfung) bereitgestellt werden, was zusätzliche Mechanismen benötigt. Des Weiteren muss gewährleistet sein, dass der nächste Schritt des komplexen Web Service bekannt ist. Damit dies der Fall ist, kann man dem Web Service einen Prozessplan mitgeben, welcher von einer intelligenten Instanz interpretiert werden kann. Dieses kann zum Beispiel mit der Hilfe der *Web Ontology Language for Services* (OWL-S) geschehen, einer Sprache, die aus dem Bereich des Semantic Web stammt. Dabei ist aber zu beachten, dass die semantischen Beschreibungen der Web Services noch weitere technische Informationen zur Verfügung stellen

müssen, damit gewährleistet wird, dass auch eine sinnvolle Wahl des gewünschten Web Services erfolgt, z. B. hinsichtlich Quality-of-Service-Kriterien [41]. Jedoch sind die Forschungen auf dem Gebiet der dynamischen Web-Service-Komposition noch nicht soweit fortgeschritten. Es gibt lediglich in einer der im folgenden Abschnitt beschriebenen Web-Service-Kompositionssprachen die Möglichkeit, diese mit den Ansätzen des Semantic Web zu verbinden.

4.2.3.2 Kompositionssprachen und –werkzeuge

Die meisten der im folgenden Abschnitt beschriebenen Kompositionssprachen erstellen eine Datei, welche einer WSDL-Datei sehr ähnlich ist und alle Informationen zum Ablauf des komplexen Dienstes beinhaltet. Dabei nutzen sie die Tatsache, dass ein einfacher Web Service durch seine WSDL-Datei eindeutig beschrieben wird und ansprechbar ist. Dadurch ist es möglich, einen einfachen Web Service in den Ablauf eines komplexen Web Services einzubinden. Die die Komposition beschreibende Datei eines komplexen Web Services kann mit Hilfe einer zur Beschreibungssprache gehörenden Engine ausgeführt werden.

4.2.3.2.1 Standard-Elemente der Kompositionssprachen

Die meisten bekannten Kompositionssprachen sind entweder vom W3C bzw. von OASIS standardisiert. Grundsätzlich besteht Einigkeit im Bereich der Web-Service-Kompositionssprachen nur über die unteren Schichten des Web-Service-Protokollstacks. Ansonsten haben diverse Firmenallianzen ihre eigenen Kompositionssprachen entwickelt, was im Bereich der Web-Service-Komposition zu einem babylonischen Sprachgewirr geführt hat.

Trotz der Sprachenvielfalt im Bereich der Kompositionssprachen haben alle ein sehr ähnliches Portfolio an Elementen, mit denen sie einen komplexen Web Service beschreiben. Im Folgenden sollen die Elemente, die in allen Sprachen gleich oder zumindest ähnlich sind, kurz erläutert werden.

- `sequence`: Die mit diesem Element gekapselten Befehle werden in einer sequentiellen Folge ausgeführt. Damit ist es möglich, mehrere unterschiedliche Web Services nacheinander anzusprechen und auszuführen.
- `flow`: Dieses Element ermöglicht im Gegensatz zu `sequence`, dass die in ihm gekapselten Befehle parallel ausgeführt werden. Es wird nicht in allen Sprachen mit `flow` bezeichnet, sondern heißt z. B. bei BPML `all`.
- `invoke`: Es startet mit der Angabe der WSDL-`portType`-Information eines Web Service einen One-Way- oder Request-Response-Nachrichtenaustausch. Somit ist dieses Element für die synchrone Kommunikation mit einem Web Service zuständig.
- `receive`: Mit diesem Element wird angegeben, dass der Business Process so lange wartet, bis er die passende Nachricht aus einem asynchronen Web-Service-Aufruf mittels `invoke` oder `reply` erhält.
- `reply`: Dieses Element erlaubt es, eine Antwortnachricht auf ein `receive` zu senden. Die Kombination dieser beiden Elemente ermöglicht einen asynchronen Request-Response-Nachrichtenaustausch über den WSDL-PortType des gewünschten Web Services.
- `assign`: Dieses Element ermöglicht eine Wertzuweisung auf eine Variable oder die Kopie einer Variablen. Dabei kann innerhalb eines `assign` mehrmals das Kindelement `copy from... to...` vorkommen, welches den Kopiervorgang durchführt.

[41] Unter anderem in [Jan03, JMP03]).

- `switch`: Man kann damit ermöglichen, dass verschiedene Pfade innerhalb eines Business Process eingeschlagen werden. Dabei wird eine bestimmte Bedingung auf `true` oder `false` getestet. Somit verhält sich dieses Element wie eine `switch case`-Anweisung in der Programmiersprache JAVA.
- `while`: Mit diesem Element kann man, wie in einer herkömmlichen Programmiersprache, eine kopfgesteuerte Schleife erstellen.
- `compensate`: Dieses Element löst eine Rücksetzungsmethode auf, welche dafür sorgt, dass der Ablauf auf einen festgelegten Punkt zurückgesetzt wird, so dass wieder Datenkonsistenz besteht.

Unternehmen sollten zur Orchestrierung von Web Services auf standardbasierte Plattformen und Werkzeuge zurückgreifen, die sich für viele Zwecke eignen. Nur eine Orchestrierungsinfrastruktur, die sich auf Standards stützt, kann die erforderliche Kompatibilität zwischen Produkten unterschiedlicher Hersteller gewährleisten, um allen Anforderungen bei unternehmensinternen und -übergreifenden Integrationen gerecht zu werden.

4.2.3.2.2 Beispiele für Kompositionssprachen

WS-BPEL (auch bekannt unter BPEL4WS)

Web Services Business Process Execution Language (WS-BPEL) wurde im Jahr 2002 noch unter dem Namen *Business Process Execution Language for Web Services* (BPEL4WS) zur Standardisierung bei OASIS eingereicht. Die Version 2.0 befindet sich im Moment im Working-Draft-Status [OAS05b]. WS-BPEL ist eine gemeinsame Entwicklung von IBM, Microsoft und BEA und stellt eine Sprache für formale Spezifikationen von Geschäftsprozessen dar. Im Rahmen der Beschreibung können Festlegungen über die Reihenfolge sowie die Abhängigkeiten und Verknüpfungen der einzelnen Dienste in dem entstehenden Prozess getroffen werden [Cla03].

WS-BPEL ist eine Weiterentwicklung der Sprachen XLANG und WSFL und hat die verschiedenen theoretischen Ansätze beider Sprachen übernommen, weshalb man mit WS-BPEL einige Standard-Sprachkonstrukte auf unterschiedliche Weise umsetzen kann. Auf Grund dieser Fusion von verschiedenen Kompositionssprachen hat WS-BPEL zwei Schwerpunkte. Der erste zielt auf die Beschreibung der Ablaufprotokolle unabhängig vom internen Ablauf der einzelnen Web Services. Ein *Abstract Process* nutzt eine Prozessbeschreibung, welche spezifiziert, wie die sichtbaren Nachrichten aller am Ablauf beteiligten Web Services aussehen. Demgegenüber gibt es noch den zweiten Schwerpunkt mit den *Executable Business Processes*, die das Verhalten eines Teilnehmers in einer laufenden Interaktion definieren. Somit vereint WS-BPEL zwei Möglichkeiten, mit denen man einen Geschäftsablauf beschreiben kann. Dabei ist *Abstract Process* an das Choreographie-Prinzip angelehnt und *Executable Business Process* spiegelt dagegen die Orchestrations-Idee wider.

Bei der Anwendung von WS-BPEL für die Komposition von komplexen Web Services ist es nicht erforderlich, dass man immer zwischen dem öffentlichen Verhalten und den internen Abläufen unterscheidet. Denn WS-BPEL hat für beide Bereiche einen gemeinsamen Kern, welcher durch die Elemente zur Erstellung von *Abstract Processes* und *Executable Business Processes* noch ergänzt wird. Dies hat den Vorteil, dass man einen komplexen Web Service implementieren und dann bei einer Weiterentwicklung ohne Probleme auch in einen größeren Geschäftsablauf integrieren kann, indem man den Bereich des *Abstract Process* im- beziehungsweise exportiert.

Um einen Web Service mithilfe von WS-BPEL in einen komplexen Business Process zu integrieren, liefert der WS-BPEL-Kern ein Interface, welches mit dem Element `partner`

`link` gekapselt wird. Dabei setzt WS-BPEL an der WSDL-Schnittstelle des Web Service an (siehe Abbildung 4-11). WS-BPEL nutzt bei der Komposition und beim Ablauf eines komplexen Web Service immer die durch die WSDL-Datei beschriebenen Datentypen des jeweiligen Web Service.

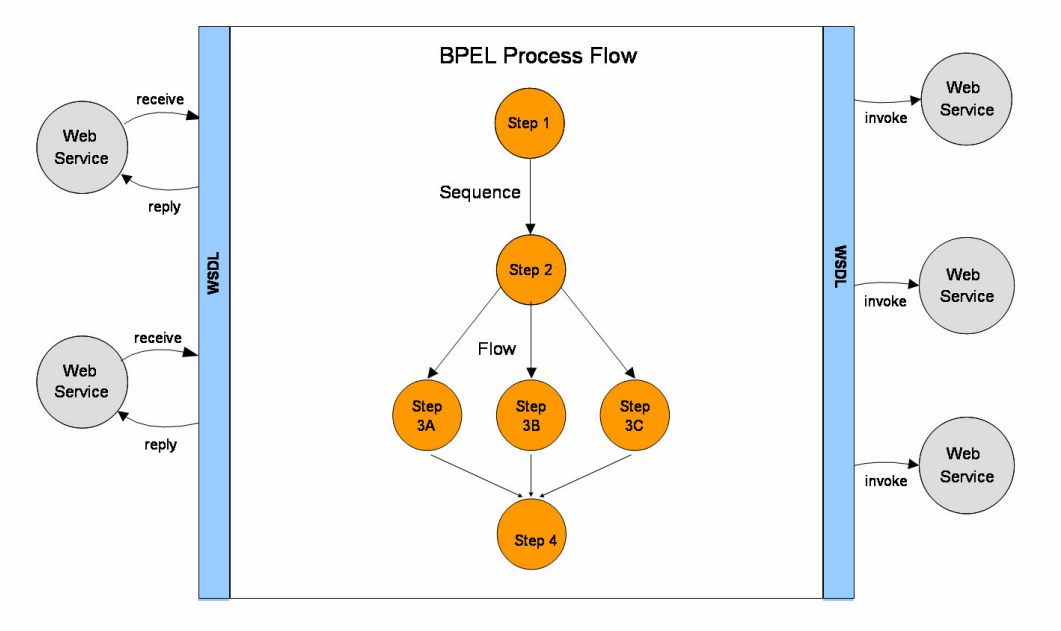

Abbildung 4-11: Funktionsprinzip einer BPEL-Komposition (in Anlehnung an [PEL03])

Die Grundbausteine der Sprache bilden so genannte Aktivitäten. WS-BPEL unterscheidet dabei zwei verschiedene Arten von Aktivitäten - die Basisaktivitäten und darauf aufbauend die strukturierten Aktivitäten.

Die **Basisaktivitäten** bilden die Grundbausteine eines Web Service und können nicht rekursiv verschachtelt werden. In WS-BPEL sind zur Zeit acht verschiedene Basisaktivitäten definiert: `<invoke>`, `<receive>`, `<reply>`, `<wait>`, `<assign>`, `<throw>`, `<terminate>` und `<empty>`. Die Aktivitäten können mit Parametern näher bestimmt werden. Sie sind äquivalent zu den in Kapitel 4.2.3.2.1 beschriebenen Standard-Elementen der meisten Kompositionssprachen. Die genaue Bedeutung jeder Basisaktivität, sowie die zulässigen Parameter sind in [CGK+02] ausführlich beschrieben.

Aufbauend auf den Basisaktivitäten stellt die Sprache **strukturierte Aktivitäten** zur Verfügung. Sie definieren die kausale Ordnung der Basisaktivitäten und können rekursiv geschachtelt werden. Definiert sind die Aktivitäten `<sequence>`, `<switch>`, `<while>`, `<pick>` und `<flow>`. Von besonderem Interesse ist die Aktivität `<flow>`, sie führt die in ihr vorhandenen Aktivitäten parallel aus. Innerhalb eines `<flow>` kann mit Hilfe von Links (`<links>`) zusätzlich eine Ordnung zwischen den Aktivitäten definiert werden. Die Möglichkeit der Definition parallelen Verhaltens erlaubt einerseits eine sehr effiziente Modellierung, sie bringt andererseits aber auch Probleme bei vorhandenen Abhängigkeiten zwischen Diensten mit sich.

Ein weiteres wichtiges Element ist `<switch>`, welches die Implementierung alternativer Abarbeitungspfade zulässt und die Entscheidung zur Laufzeit von einer Bedingung abhängig macht, die mittels `<case condition>` festgelegt werden kann. Dies ermöglicht ein generisches Design der BPEL-Prozesse und erhöht die Wiederverwendbarkeit der entstehenden komplexen Web Services.

Das folgende Listing 4-3 zeigt die Beschreibung einer sequentiellen Aktivität, die einige Basisaktivitäten einschließt. Das Beispiel zeigt ein typisches Einkaufsszenario aus der Sicht des Anbieters. Er empfängt eine Nachricht vom Käufer, ruft mit `<invoke>` einen anderen Service auf und sendet mit `<reply>` eine Antwort an den Aufrufenden zurück.

```
<sequence>
  <receive partner="buyer" ... operation="sendOrder" container="request"/>
  <invoke partner="supplier" ... operation="request" container="order"/>
  <reply partner="buyer" ... operation="response" container="proposal"/>
</sequence>
```

Listing 4-3: Beispiel für WS-BPEL [Pel03]

Die Attribute `container` und `partner` sind zwei wichtige Elemente in WS-BPEL. `Container` identifiziert den Datenaustausch in einem Nachrichtenfluss, indem es auf einen WSDL `messageType` referenziert. Sie werden außerdem für die persistente Datenhaltung während einer Web-Service-Anfrage eingesetzt. Ein so genannter `partner` kann ein beliebiger Web Service sein, der vom Prozess eingebunden wird. Dabei wird jeder Partner auf eine spezielle Rolle abgebildet, die er im Geschäftsprozess einnimmt.

Zusätzlich verfügt WS-BPEL über eine ausgeprägte Fehler- und Transaktionsverwaltung, die auf die Spezifikationen *WS-Coordination* und *WS-Transaction* aufbauen. In WS-BPEL liegen der Transaktionsmechanismus und die Fehlerbehandlung eng nebeneinander. Das Element `<scope>` gibt die Möglichkeit, eine Menge von Aktivitäten zu einer Transaktion zu gruppieren. Innerhalb dieser Gruppierung können vom Entwickler Operationen definiert werden, die bei einem Fehler ausgeführt werden. Das bedeutet, dass die Aktivitäten innerhalb eines `<scope>` entweder vollständig oder gar nicht ausgeführt werden. Diese Mechanismen sind auch zwingend notwendig, um einen Geschäftsprozess effektiv zu koordinieren und zu verwalten.

WS-BPEL ist gegenwärtig der beste standardisierte Ansatz für die Komposition von Web Services, da es, wie erläutert, neben dem *Executable Business Process* auch die Möglichkeit der *Abstract Processes* bietet. Diese können erst zur Laufzeit im Zusammenwirken mit anderen Diensten ausführbar oder als logische Grundstruktur universell verwendet werden. Durch die Zusammenarbeit von Microsoft und IBM konnten das graphisch-orientierte WSFL und das eher ablauf-orientierte XLANG verknüpft und die Vorteile von beiden Ansätzen in WS-BPEL integriert werden. Außerdem bietet WS-BPEL auf Grund seiner Gliederung in einen Teil mit den Kernelementen und zwei Erweiterungsbereiche eine gute Ausgangsposition, um die Sprache durch sinnvolle Elemente zu ergänzen.

WSCI

Web Service Choreography Interface (WSCI) [sprich: Whisky], welches im Jahr 2002 veröffentlicht wurde [W3C02b], wird oft als Konkurrent von WS-BPEL gesehen. Dieser Standard wurde vor allem unter Mitarbeit von SUN, Intalio, SAP und BEA unter dem Dach des W3C entwickelt. Die XML-basierte Sprache WSCI bietet als Lösung zur Komposition von Web Services die Nutzung eines Interfaces an, welches das Verhalten des Web Service eindeutig beschreibt. Dadurch sollen Web Services zusammenarbeiten können, ohne dass diese vorher in einen *Business Process* integriert wurden. WSCI setzt dabei konsequent die Idee der Web-Service-Choreographie um.

Grundsätzlich wird mit WSCI nur das sichtbare und öffentliche Verhalten von Web Services beschrieben. Der Nachrichtenaustausch zwischen den beteiligten Diensten steht dabei im Vordergrund. Man kann WSCI als eine Erweiterung von WSDL betrachten. WSDL wird dabei nicht qualitativ oder quantitativ bereichert, sondern von WSCI umschlossen und so für die Web-Service-Choreographie nutzbar gemacht (siehe Abbildung 4-12). Dabei

beschreibt WSDL die Schnittstellen, um einen Web Service zu nutzen, und WSCI definiert, wie die Interaktion zwischen diesem und einem weiteren Web Service ablaufen kann.

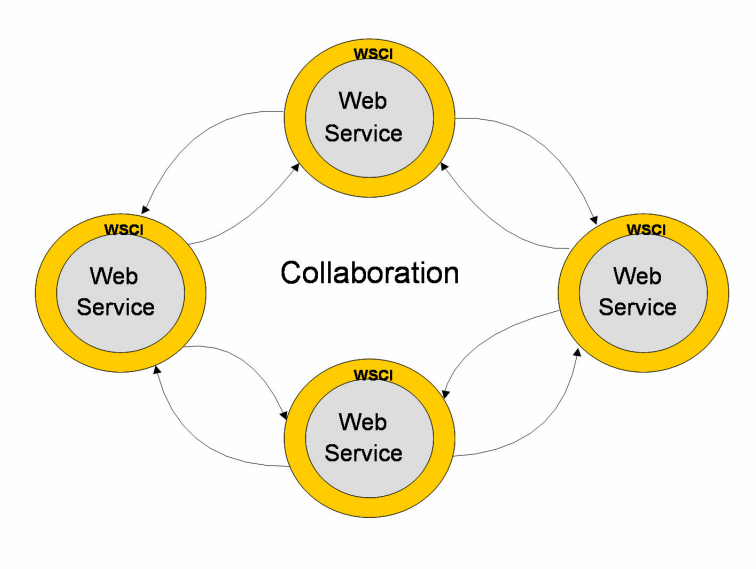

Abbildung 4-12: WSCI-Funktionsprinzip [PEL03]

WSCI unterstützt ebenfalls Basis- und strukturierte Aktivitäten.

Basisaktivitäten: Das `<action>`-Element beschreibt die grundlegende Anfrage- bzw. Antwortnachricht. Jede Aktivität definiert dabei eine WSDL-Operation und die damit verbundene Rolle des jeweiligen Teilnehmers. Ein externer Service kann mittels des `<call>`-Elements eingebunden werden.

Strukturierte Aktivitäten: Hier existieren eine Vielzahl von Aktivitäten, einschließlich der sequenziellen und parallelen Prozessausführung, sowie bedingte Schleifen (analog zu WS-BPEL).

Zusätzlich führt WSCI noch die Aktivität `<all>` ein, die verwendet wird, um die Ausführung von Aktionen ohne Berücksichtigung einer bestimmten Reihenfolge zu definieren. Listing 4-4 zeigt ein einfaches Beispiel einer WSCI-Beschreibung. Der abgebildete Einkaufsprozess ist eine Anordnung von zwei sequentiell auszuführenden Aktivitäten namens `ReceiveOrder` und `Confirm`. Jede Aktivität wird durch das Schlüsselattribut `operation` auf einen WSDL `portType` abgebildet. Das Element `<correlation>` erzeugt eine Wechselbeziehung zwischen den beiden Aktionen.

```
<process name="Purchase" instantiation="message">
  <sequence>
    <action name="ReceiveOrder" role="Agent" operation="tns:Order">
    </action>
    <action name="Confirm" role="Agent" operation="tns:Confirm">
      <correlate correlation="tns:ordered"/>
      <call process="tns:Purchase"/>
    </action>
  </sequence>
</process>
```

Listing 4-4: Beispiel für WSCI [Pel03]

Weiterhin unterstützt WSCI auch *Business Transactions* und *Exception Handling*. Es ist möglich, einen speziellen Kontext innerhalb von WSCI-Aktivitäten zu definieren, so dass im Fehlerfall das Ergebnis dieses Abschnittes durch ein `rollback` wieder zurückgesetzt werden kann.

BPML

Die *Business Process Management Language* (BPML) ist eine Metasprache, die die Abbildung von Geschäftsprozessen bei der Komposition von komplexen Web Services in den Vordergrund stellt. Sie wurde anfänglich entwickelt, um Geschäftsprozesse zu modellieren und in einem *Business Process Management System* (BPMS) auszuführen. Seit dem Jahr 2000 wird BPML von der *Business Process Management Initiative* (BPMI.org) entwickelt, einer unabhängigen Organisation, die weltweit von den großen IT- und Beratungsunternehmen unterstützt wird. Der erste Entwurf dieser Spezifikation wurde im März 2001 veröffentlicht [BPM01] und beinhaltete das WSCI-Protokoll für die Beschreibung der öffentlichen Interaktionen und Choreographien. WSCI wird jetzt als ein Teil von BPML betrachtet, der die Interaktionen zwischen den Web Services definiert, und BPML definiert den Geschäftsprozess, der sich hinter jedem Web Service befindet.

BPML kann aber auch mit BPEL verglichen werden, weil es ähnliche Konstrukte und Aktivitäten für den Prozessfluss zur Verfügung stellt [Pel03]. Es sind sowohl Basisaktivitäten für das Senden, Empfangen und Aufrufen von Services als auch strukturierte Aktivitäten vorhanden. Zusätzlich unterstützt BPML die Taskverwaltung, transparente Persistenz von Objekten, Wechselbeziehungen zwischen Instanzen und rekursive (De)Komposition. Die rekursive Komposition schafft die Möglichkeit, Unterprozesse in große Geschäftsprozesse einzubinden. Auch Transaktionsunterstützung und Fehlerbehandlungsmechanismen sind vorhanden, wobei durch Kompensationstechniken auch komplexe Transaktionen gewährleistet werden können. Diese Kompensationstechniken verwenden das `scope`-Element ähnlich wie BPEL zuzüglich der Fähigkeit der Schachtelung von Transaktionen, was mit BPEL bisher nicht möglich ist. Trotz einiger Vorteile wird BPML seit einiger Zeit nicht mehr weiterentwickelt und im Standard-BPM-Stack durch BPEL ersetzt.

4.2.3.2.3 Kompositionswerkzeuge

Es ist mittlerweile eine große Palette von Kompositionswerkzeugen auf dem Markt verfügbar. Jedoch sind die Ausführungsumgebungen nicht standardisiert und zertifiziert, sondern werden von den verschiedenen Herstellern oder Open-Source-Entwicklergruppen implementiert. Deren Grundlage ist der Standard, der aber je nach Zielvorstellung mehr oder weniger komplett umgesetzt wird. Daher ergibt sich für den Entwickler komplexer Web Services ein unüberschaubarer Markt, auf welchem es unterschiedliche Tools für die gleiche Aufgabe gibt.

Allen Produkten sind jedoch grundsätzliche Komponenten gemein. Dazu gehört erstens eine server-basierte *Orchestration Engine* zur Verbindung von Applikationen, Prozessen und Daten. Zweitens gibt es immer eine mehr oder minder umfassende Palette von Konnektoren für den Anschluss an diverse Anwendungs-, Plattform-, Middleware-, Transport- und Datensysteme. Und drittens gehören meist visuelle Tools zum Angebotsumfang, die eine graphische Schnittstelle zur Definition, Abbildung, Umwandlung und Verwaltung sowie zum Reporting und zur Analyse von Orchestrierungen anbieten. Die folgende Übersicht (Tabelle 4-1) über derzeit am Markt vorhandene Kompositionswerkzeuge wurde [Hof05] entnommen.

Grundsätzlich unterstützen alle beschriebenen Werkzeuge den Entwickler bei der Komposition von Web Services. Außer dem WSCI-Editor unterstützen alle die Sprache WS-BPEL, was dafür spricht, dass sich WS-BPEL auf dem Markt bisher durchsetzen konnte. Hervorzuheben ist der Oracle *Process Manager*, der mit dem BPEL-Designer ein visuelles Werkzeug zum Erstellen von komplexen Web Services anbietet, das durch seine gute Bedienbarkeit und Einfachheit besticht. Deshalb wurde er auch für die praktische Umsetzung von Web-Service-Kompositionen innerhalb der vorliegenden Arbeit genutzt.

Tool	Hersteller	unterstützte Sprache	Kompositions-schwerpunkt	Design Plugin / unabhängiges Werkzeug	benötigte Server	Open-source
BPWS4J	IBM	BPEL 1.1	Orchestration	+/ Eclipse	Tomcat/ WebSphere	-
Service Orchestrator	OpenStorm	BPEL 1.1	Orchestration	+	Tomcat/ WebSphere / IIS	-
Twister	Sourceforge	BPEL 1.1	Orchestration	-	Apache Tomcat	+
ActiveBPEL	Active Endpoints	BPEL 1.1	Orchestration	+/ Eclipse, .NET Studio	Jboss/Web Sphere/ WebLogic	Engine + Tool -
Process Manager	Oracle	BPEL 1.1	Orchestration	+/ Eclipse	Oracle 10 g / Jboss/ WebLogic	-
WSCI Editor	SUN	WSCI 1.0	Choreographie	+	-	-

Tabelle 4-1: Übersicht über Kompositionswerkzeuge

Eine detaillierte Beschreibung der einzelnen derzeit am Markt befindlichen Produkte und Werkzeuge ist in [Hof05] nachzulesen.

4.2.3.3 Fazit

Zur Orchestration komplexer Dienste hat sich in den letzten Jahren die Meta-Sprache WS-BPEL durchgesetzt. Durch die breite Unterstützung namhafter Hersteller entstand mittlerweile eine Vielzahl an Beschreibungs- und Design-Werkzeugen sowie Ausführungsumgebungen für WS-BPEL, was ebenfalls zum weiteren Ausbau seiner Vormachtstellung beitrug. WS-BPEL ermöglicht die Orchestrierung beliebiger Web Services zu einem *Business Process*, wobei dieser wiederum als eigenständiger Web Service zur Verfügung gestellt wird. Somit ist die Abbildung von komplexen Workflows und Geschäftsprozessen möglich. Durch die Einführung komplexer zusammengesetzter Services kann die Anzahl und Komplexität der Schnittstellen zwischen den einzelnen Anwendungssystemen deutlich verringert werden.

Um eine dynamische Integration der Web Services zu ermöglichen, kann man auch abstrakte Prozesse in BPEL beschreiben, die noch keine Bindung an konkrete Web Services enthalten, sondern die benötigten Dienste nur abstrakt beschreiben. Dies ermöglicht eine flexiblere Einbindung unterschiedlicher Dienste, die die gleiche Funktionalität anbieten. Die Bindung an konkrete Dienste erfolgt erst, wenn ein ausführbarer Prozess (*Executable Process*) erzeugt wird.

4.2.4 Suche geeigneter Dienste

Zur Suche geeigneter Dienste kann auf die Funktionalität des UDDI-Registers zurückgegriffen werden. Grundsätzlich gibt es zwei Möglichkeiten, die UDDI-Suchfunktionen zu nutzen. Der Nutzer kann direkt über ein Web-Interface im Register suchen und erhält eine Liste verfügbarer Web Services mit zugehöriger Quelle (URI) und Schnittstellenbeschreibung (WSDL) zurück. Um eine Automatisierung der Suche überhaupt erst zu ermöglichen, wurde die UDDI-API geschaffen, deren Funktionsweise im Folgenden näher erläutert wird.

Die im UDDI-Register veröffentlichten Web Services können lediglich über die Eingabe einfacher Suchbegriffe gefunden werden, was der Heterogenität von Web Services nur ungenügend Rechnung trägt. Deshalb sind zusätzliche semantische Beschreibungen der Schnittstellen und des Funktionsumfanges von Web Services notwendig, um die Suche effizienter zu gestalten. Die dafür nutzbaren Ansätze aus dem Bereich des Semantic Web werden ebenfalls in diesem Kapitel erläutert.

4.2.4.1 Nutzung der UDDI-API

Die Anfrage-API von UDDI unterstützt eine Untermenge der Funktionalität des SOAP-Protokolls. So können zum Beispiel keine Header-Bereiche definiert werden. Enthalten sind grundlegende Operationen zum Einfügen, Ändern und Auffinden von Daten. Die UDDI-API lässt sich in die Inquiry-API, über die man Anfragen zum Auffinden von Metadaten anhand bestimmter Selektionskriterien tätigen kann, und die Publishing-API, mit der sich Einträge registrieren, verändern und löschen lassen, unterteilen.

Zum automatisierten Auffinden von Diensten gibt es die folgenden Standard-API-Funktionen:

```
bindingTemplateList:find_binding
      (maxRows, serviceKey, findQualifiers, categoryBag, tModelBag)
businessList:find_business
      (maxRows,   findQualifiers,   identifierBag,   name,   categoryBag,
      tModelBag)
serviceList:find_service
      (maxRows,   businessKey,   findQualifiers,   name,   categoryBag,
      tModelBag)
tModelList:find_tModel
      (maxRows, findQualifiers, name, identifierBag, categoryBag)
bindingDetail:get_bindingDetail
      (bindingKey)
businessDetail:get_businessDetail
      (businessKey)
businessDetailExt:get_businessDetailExt
      (businessKey)
serviceDetail:get_serviceDetail
      (serviceKey)
tModelDetail:get_tModelDetail
      (tModelKey)
```

In den Klammern sind die Namen von Elementen und Attributen aufgeführt, die bei der jeweiligen Methode dazu verwendet werden können, um die Suche einzugrenzen. `maxRows` kann als Attribut innerhalb des Methodenelementes verwendet werden, um die Ergebnismenge in ihrer Anzahl einzuschränken, `businessKey` und `serviceKey` schränken die Suche auf bestimmte Schlüsselwerte ein. Über das `name`-Element wird innerhalb der entsprechenden Methode die Suche auf Dienste, Anbieter oder `tModels` mit einem bestimmten Namen eingeschränkt, `categoryBag` grenzt die Suche anhand von Kategorien ein und `tModel-Bag` unterstützt die Suche nach Diensten, die eine bestimmte Schnittstelle implementieren.

Daraus ergeben sich die folgenden zwei Möglichkeiten zum Auffinden von Web Services über UDDI:

- direkte Suche nach Web Services,
- Suche nach Schnittstellen und anschließend Suche nach Web Services, die diese Schnittstelle implementieren.

Im ersten Fall werden die Dienste im Allgemeinen zuerst über die `find_business`-Anfrage ausfindig gemacht und nach den gewünschten Kriterien eingeschränkt. Da dabei nicht davon auszugehen ist, dass nur ein Dienst in der Antwortliste zurückgeliefert wird, lässt sich das Ergebnis über eine weitere `find_service`-Anfrage (siehe Listing 4-5), die für jeden zurück gelieferten Dienst gestartet werden muss, weiter einschränken.

```
<find_business maxRows="10" generic="2.0" xmlns="urn:uddi-org:api_v2">
  <findQualifiers>
    <findQualifier>caseSensitiveMatch</findQualifier>
  </findQualifiers>
  <name>%IBM%</name>
  <name>%SAP%</name>
</find_business>
```

Listing 4-5: `Find_service`-Anfrage an UDDI-Registry

Die zweite Variante ist von zentralerer Bedeutung, wenn Web Services mit gleicher Funktionsweise, also gleichem `tModel`, gesucht werden sollen. Dabei wird zuerst manuell über die `find_tModel`-Anfrage, die zum Beispiel auch durch die UDDI-Browser bereitgestellt wird, die gewünschte Web-Service-Schnittstelle gesucht. Daraus kann dann der zugehörige `tModelKey` (eindeutiger Bezeichner) entnommen werden und für eine `find_business`-Anfrage als Inhalt des `tModelBag`-Elementes verwendet werden. Das Ergebnis dieser Anfrage ist dann eine Liste (`businessList`) mit verkürzten `BusinessEntity`-Informationen (`businessInfos`) von Anbietern, welche das angegebene `tModel` implementieren.

In UDDI Version 3 wurden die Anfragemöglichkeiten durch neue Suchbezeichner (`findQualifier`), zusammengesetzte Anfragen und die intelligentere Behandlung von großen Ergebnismengen wesentlich verbessert.

Die wichtigsten Erweiterungen der Suchbezeichner sind:

- Übereinstimmung (un)abhängig von Groß- und Kleinschreibung,
- Sortierung (un)abhängig von Groß- und Kleinschreibung,
- ungefähre Übereinstimmung (vergleichbar mit der SQL-LIKE-Syntax),
- binäre Sortierung,
- exakte Übereinstimmung,
- vorhandene Signatur.

Viele große IT-Unternehmen betreiben UDDI-Server zur Lokalisierung von Web Services. Leider sind diese Dienste aber entweder kostenpflichtig oder enthalten nur schwach besetzte, fehlerhafte Einträge. So finden sich z.B. auf der Test-Registry von IBM[42] nur sehr wenige nutzenswerte Einträge.

Für eine vollautomatische Kommunikation zwischen Anwendungen ist der pure Einsatz von UDDI und WSDL meist aber nicht ausreichend. So liefern t-Models Vorbedingungen für einen Web Service, nicht jedoch für sämtliche darin definierten Operationen. Auch sind die Variablennamen in den WSDL-Spezifikationen oft nicht eindeutig semantisch belegt. So stößt man bei der Suche nach geeigneten Web Services schnell an die Grenzen der aktuellen Standards. Zum einen gibt es das Problem der syntaktischen Heterogenität der Schnittstellen zum anderen sind die Möglichkeiten, zusätzliche Informationen über einen Dienst bereitzustellen, relativ gering. Somit sind auch Such- und Auswahlkriterien im UDDI nur auf eine Schlüsselwortsuche in den festgelegten Metadaten begrenzt. Die Suche nach funktionsgleichen Diensten ist dadurch nicht in dem Maße erfolgreich, wie man sich das wünschen würde. Falls eine Menge funktionsgleicher Dienste im UDDI vorliegt, könn-

[42] http://www-3.ibm.com/services/uddi/testregistry/inquiryapi

ten erweiterte Suchkriterien dazu dienen, die Ergebnisse besser einzuschränken und die Auswahl des geeigneten Dienstes zu unterstützen.

4.2.4.2 Semantische Beschreibung der Dienste

Die Standard-Protokolle zur Beschreibung von Web Services fokussieren hauptsächlich die syntaktischen Aspekte einer SOA und erfüllen die Anforderungen an eine flexible Kopplung von Diensten nicht ausreichend. Zur vollständigen Automatisierung der Dienstsuche und -integration in die Arbeitsumgebung muss jedoch die semantische Komponente hinzukommen. Das flexible Lösen einer Aufgabe setzt voraus, dass die Bedeutung (Semantik) der Aufgabe und des Services eigenständig (automatisch) interpretiert werden kann. Ohne diese Möglichkeit muss jeder Schritt und jeder zu nutzende Service explizit vorgegeben werden [DoJ04]. Woher weiß eine Anwendung, welchen Service sie nutzen kann, um ein gesetztes Ziel zu erreichen?

Im UDDI ist das Auffinden eines bestimmten Web Service lediglich über eine bloße Schlüsselwortsuche realisiert. Für eine Automatisierung ist dies i. d. R. nicht ausreichend, da die Begriffe des Alltags nicht immer eindeutig bzw. kontextabhängig sind. So kann man sich vorstellen, dass Benutzer bei der Eingabe von Suchkriterien verschiedene Synonyme für einen Terminverwaltungsdienst wie z.B. „Kalender“ oder „Terminplaner“ verwenden. Deshalb bieten neuere UDDI-Implementierungen schon die Möglichkeit, auch Web-Service-Beschreibungen explizit mit Bedeutung zu versehen.

Eine vollständig automatisierte Nutzung von Web Services wird jedoch erst dann möglich sein, wenn alle Komponenten in der Lage sind, Semantik in einer maschinenlesbaren Form abzulegen, zugreifbar zu machen und gegebenenfalls auszuwerten. Bei der Erweiterung von Web Services um Semantik verfolgt die Web-Service-Gemeinde den bisher erfolgreichen Weg, bereits existierende Standards, die eine gewünschte Funktionalität bieten, zu integrieren. Deshalb werden die bekannten Standards des *Semantic Web* wie RDF (*Resource Description Framework)*, RDF-Schema und die Verwendung von Ontologien für die semantische Beschreibung von Web Services genutzt.

Es gibt verschiedene Ansätze zur semantischen Beschreibung von Diensten, welche da wären:

- RDF-S (*Resource Description Framework - Schema*)
- SWSL (*Semantic Web Service Language*), WSMF (*Web Service Modeling Framework*)
- OWL-S (*Web Ontology Language for Services*), DAML-S (*DARPA Agent Markup Language for Services*)
- *Topic Maps*

Exemplarisch sollen RDF-S und OWL-S im Rahmen dieser Arbeit näher untersucht werden, da sie vom W3C standardisiert wurden und im Semantic-Web-Bereich meist Verwendung finden.

4.2.4.2.1 RDF-S (Resource Description Framework - Schema)

Grundidee des *Resource Description Framework* (RDF) ist, dass Ressourcen durch ihre Eigenschaften beschrieben werden. Das RDF-Datenmodell besteht aus drei Elementtypen:

Subjekte: werden von einem URI beschrieben.

Prädikate: definieren eine binäre Relation zwischen Subjekten und Objekten.

Objekte: spezifizieren den Wert eines Prädikats.

Alle drei zusammen ergeben ein RDF Statement, das an einem Beispiel in Abbildung 4-13 veranschaulicht werden soll.

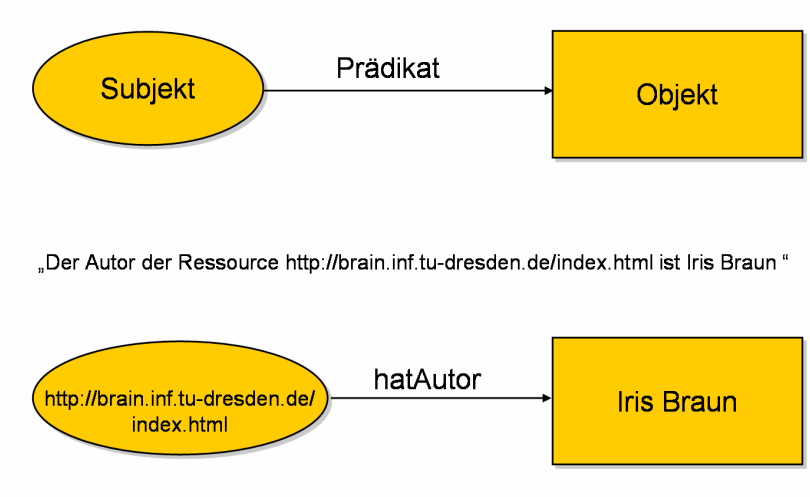

Abbildung 4-13: RDF-Statement

RDF-Schema (RDFS) bündelt diese Metadaten und beschreibt die Bedeutung, die Eigenschaften und die Beziehungen zwischen Ressourcen. RDF-Schemata können außerdem ihre Eigenschaften vererben.

Es gibt drei Klassen von Schemata:

- *Resources*: die Klasse aller Ressourcen,
- *PropertyType*: die Klasse aller Prädikate,
- *Class*: die Klasse aller möglichen Werte.

RDF-S gestattet es, die Inhalte einer RDF-Beschreibung semantisch zu spezifizieren, ohne Aussagen über deren lexikalische Darstellung zu treffen. Somit ermöglicht es die Einführung eines domänenspezifischen Vokabulars. Ob verschiedene Domänen ähnliche Vokabulare für die gleiche semantische Beschreibung eines Dienstes benutzen, kann dagegen in RDF-S nicht überprüft werden. Um einen Abgleich von Aussagen verschiedener Domänen durchführen zu können, benötigt man so genannte Ontologien - Namensräume mit Konzepten und Aussagen über deren Zusammenhänge.

4.2.4.2.2 OWL-S (Web Ontology Language for Services)

Eine weitere Methode zur semantischen Beschreibung von Web Services ist die Nutzung von so genannten Ontologien und einem darauf aufbauenden System zur Interpretation und Weiterverarbeitung dieser Informationen. Eine Ontologie definiert einen Namensraum (Domäne) und besteht aus einer Fülle von Wissensbegriffen (Vokabular), semantischen und syntaktischen Beziehungen sowie bestimmten Logikregeln dieser Domäne. Die Elemente (Konzepte) dieses Namensraums werden in der Ontologie mittels Aussagen wie „Konzept A bedeutet dasselbe wie Konzept B“ (*equivalentTo*) oder „Konzept A ist eine Spezialisierung von Konzept B“ (*instanceOf*) in Beziehung zueinander gesetzt. Die Benennung von Funktionsparametern legt für einen Benutzer eines Web Service zwar eine semantische Bedeutung nahe, für eine maschinelle Auswertung sind sie jedoch nicht ausreichend, da es sich für einen Computer nur um beliebige Strings handelt, auf deren Basis kein Matching möglich ist. Erst die Zuweisung eines Konzeptes zu einem Parameter, das „Wissen“ über die Beziehungen von Konzepten untereinander und die Verwendung der gleichen Ontologie für verschiedene Web Services des gleichen Aufgabenbereiches sorgt

für eine Vergleichbarkeit und ein darauf aufbauendes automatisiertes, semantisches Matching.

Die häufigste Form von Ontologien im Internet sind Taxonomien. Diese definieren Klassen von Objekten und deren Beziehungen untereinander. Die Strukturierung erfolgt baumartig unter Verwendung einer Rangordnung und der beschriebenen Verknüpfungen. Eine solche Ontologie ist OWL-S[43] (*Web Ontology Language for Services*), die auf RDF und XML basiert. Ein Dienst wird in OWL-S wie in Abbildung 4-14 dargestellt. Es gibt eine `Resource`, die den Dienst zur Verfügung stellt. Diese `Resource` wird standardmäßig durch einen URI beschrieben. Die OWL-S-Ontologie besteht aus den drei Teilen: `ServiceProfile`, `ServiceModel` und `ServiceGrounding`.

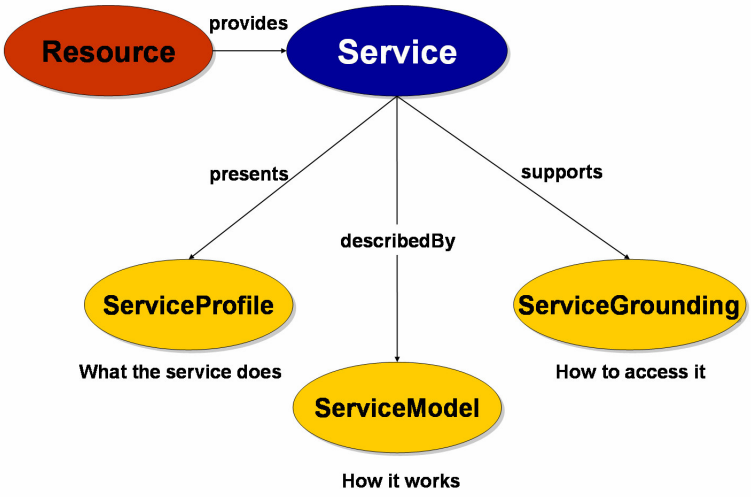

Abbildung 4-14: Top-Level der Service-Ontologie in OWL-S

Der `ServiceProfile`-Teil der Beschreibung liefert einem Agenten, der einen Web Service sucht, Informationen darüber, welche Dienstleistung der Web Service erbringt. Es beschreibt die Fähigkeiten und Parameter des Services und die Anforderungen an den Agenten. Es besteht aus zwei Teilen - der Beschreibung des Dienstes und des Anbieters und der Beschreibung des funktionalen Verhaltens des Web Service.

Das `ServiceModel` wird von Agenten benutzt, um zu überprüfen, ob der Service die gestellten Anforderungen erfüllt und um eine neue Service-Beschreibung aus mehreren Services zu komponieren. Das Servicemodell wird vom Prozessmodell erweitert, welches die Ein- und Ausgänge abstrakt modelliert. Der `Process`-Teil enthält noch etwas umfangreichere Informationen über die Funktionsweise des Dienstes.

Das `Service-Grounding` definiert, wie abstrakte Inputs und Outputs in echte *Message calls* umgesetzt werden. Es dient dem Mapping der abstrakten Service-Beschreibung auf WSDL-Bindings.

Mit der Weiterentwicklung von DAML-S zur Version 1.0 wurde die Ontologie in OWL-S umbenannt. Die grundlegende Struktur der semantischen Beschreibungen von Web Services, nämlich die Einteilung in Service-, Profile- und Grounding-Anteil blieb beim Versionswechsel von DAML 0.9 zu OWL 1.0 erhalten. Der strukturelle Aufbau der Anteile ist derzeit jedoch noch teilweise erheblichen Änderungen unterworfen.

[43] OWL-S wurde früher mit DAML-S (DARPA Agent Markup Language for Services) bezeichnet. Mit der Weiterentwicklung von DAML-S 0.9 zur Version 1.0 wurde die Ontologie in OWL-S umbenannt.

4.2.4.3 Fazit

Mit UDDI wurde ein Protokoll geschaffen, mit dem man Web Services und ihre wichtigsten Eigenschaften in zentralen Registern veröffentlichen und suchen kann. Bisher steht einer Automatisierung von Such- und Auswahlprozessen aber die syntaktische und semantische Heterogenität von Web Services im Wege, die mit derzeitigen Mitteln in UDDI nicht abgebildet werden kann. Somit ergibt sich die Notwendigkeit, dass Dienste unter verschiedensten Aspekten semantisch beschrieben werden müssen. Durch die Verwendung von maschinenverarbeitbaren Semantiken kann die Suche, die Komposition und die Ausführung von Web Services automatisiert werden.

Um semantische Gemeinsamkeiten auszudrücken, werden Ontologien verwendet, anhand derer sich Begriffe und deren Beziehungen zueinander semantisch repräsentieren lassen. Außerdem werden darüber die Bedeutungen von Termen zwischen verschiedenen Dialekten hergestellt. Es lässt sich somit prüfen, ob verschiedene Begriffe semantisch übereinstimmen. Eine solche Ontologie ist OWL-S, welche auf RDF basiert und die semantische Beschreibung von Service-Eigenschaften ermöglicht.

Um diese semantischen Informationen für die Suche von Web Services nutzen zu können, wird ein intelligenter Algorithmus benötigt, der nach semantischen Übereinstimmungen zwischen Anforderungen des Dienstnutzers und Fähigkeiten der registrierten Web Services suchen kann. Auf der Basis von OWL-S wurde an der Carnegie Mellon University in Pittsburgh dafür bereits ein Matchmaker-Modul für UDDI entwickelt [SPS04], das diese Aufgabe übernehmen kann und eine Erweiterung der Web-Service-Architektur darstellt. Dieses Modul kombiniert die semantikbasierte Suche nach Übereinstimmungen und die Suche nach Fähigkeiten von Web Services. Es ist vollständig in UDDI eingebettet und kann OWL-S-Beschreibungen beim Veröffentlichen und Suchen verarbeiten. Wie dieses Modul bei der Suche nach verfügbaren Telearbeitsdiensten genutzt werden kann, wird in Kapitel 5.3.2 detaillierter beschrieben.

Neben dem OWL-S-Matchmaker der Carnegie Mellon University gibt es zwei weitere Ansätze, die eine Eignungsprüfung von OWL-S-Beschreibungen vornehmen können: der *OWL-S Matcher* (OWLSM)[44] von der TU Berlin und der *Hybrid OWL-S Web Service Matchmaker* (OWLS-MX)[45] vom Deutschen Forschungszentrum für Künstliche Intelligenz Saarbrücken. Für beide liegen freie JAVA-Implementierungen vor. Die Integration in das UDDI-Verzeichnis wurde aber in diesen Ansätzen nicht umgesetzt.

4.2.5 Nutzerschnittstellen für Web Services

Da Web Services originär für die Kommunikation und Integration von Anwendungen und nicht für die direkte Interaktion mit Nutzern gedacht waren, thematisieren die Standards hauptsächlich die Beschreibung der Datenstrukturen, Schnittstellen und Austauschformate. Die Beschreibung einer Benutzeroberfläche (UI - *User Interface*) für Web Services wurde hingegen bisher nicht standardisiert. Wenn Web Services aber automatisch in ein Portal oder andere Webseiten integriert werden sollen, muss dem Nutzer ein Interface zur Interaktion mit dem Dienst zur Verfügung gestellt werden. Es gibt bisher einige Forschungsarbeiten auf diesem Gebiet, die bisher aber keine standardisierte Lösung anbieten können. Deshalb wurde im Rahmen dieser Dissertation untersucht, wie man Nutzerschnittstellen für Web Services beschreiben und erzeugen kann.

[44] http://owlsm.projects.semwebcentral.org/

[45] http://owls-mx.projects.semwebcentral.org/

Die von Web Services angebotene formale Beschreibung (WSDL) enthält Nachrichten- und Typinformationen für eine einfache bidirektionale Kommunikation, d.h. für eine Anfrage und das dazugehörige Ergebnis. Es können dabei sowohl HTTP-Anfragen (über GET oder POST) oder SOAP-Anfragen definiert sein. Auch ein so genannter *Access Point,* ein existierender HTML-Zugang, kann optional angegeben werden.

Um aber eine einheitliche Eingabeschnittstelle für alle Web Services zu bekommen, kann man die Typinformationen der WSDL auslesen und lokal (z.B. in einem Portlet) in HTML umwandeln. Dafür gibt es prinzipiell mehrere Möglichkeiten:

- Transformation der WSDL mit Hilfe von XSLT (*Extensible Stylesheet Language Transformation*)
    - serverseitig, d.h. im Portal mit Java-XSLT-Klassen,
    - clientseitig, d.h. im Browser,
- Nutzung eines standardisierten Prinzips wie WSUI (*Web Service User Interface*)[46],
- Erstellung und Nutzung einer Java-Klasse zur Anzeige der Ein- und Ausgabeparameter.

Die XSLT-Transformation im Browser ist sehr fehleranfällig und erfordert einen modernen Browser. Die bessere Option wäre die serverseitige Wandlung (oder zumindest Vorbehandlung) der Namespace-lastigen WSDL-Daten. Im Rahmen dieser Arbeit wird die letzte Variante implementiert (siehe Kapitel 6.1.2.3). Dabei wird aus den Typangaben der WSDL ein HTML-Eingabeformular generiert, um die Eingabeparameter an den Web Service zu übergeben. Das zurück gelieferte Ergebnis wird ebenfalls in einem HTML-Formular ausgegeben.

Da die Umwandlung in HTML-Formulare eigentlich nur für einfache Datentypen und simple Operationen erfolgen kann, wurde an der University of Stanford das WSGUI-Konzept entworfen [KKM03]. Dieses baut auf XSLT auf, verwendet aber dafür einen zugrunde liegenden GUIDD (*Graphical User Interface Deployment Descriptor*), welcher manuell für jeden Web Service erstellt werden muss. Im Produktionsbetrieb kann es jedoch auch durch einen Webserver generiert werden, der zwischen der Client-Anwendung (z.B. Browser) und dem Web Service liegt und über HTTP bzw. SOAP die Nachrichten transparent in beide Richtungen weiterreicht.

Neben der reinen Darstellung der Eingabeformulare bietet das Konzept weitere Vorteile. So werden nicht die internen WSDL-Variablennamen bei der Anzeige der Eingabe-Parameter verwendet, sondern Umsetzungen oder gar Übersetzungen auf menschenlesbare Namen vorgenommen. Eingaben können in mehreren Schritten erfolgen, inklusive der Rückkehr zum vorherigen Menü mit Erhaltung des Kontextes der bisherigen Eingaben. Schließlich werden noch dynamische Listen, vorgefüllte Auswahlfelder und einfache Filteroperationen unterstützt. Auf dieses Konzept wurde bei der weiteren Implementierung im Rahmen dieser Arbeit aufgebaut (siehe Kapitel 6.1.2.3).

Ein Konzept, welches sich verstärkt auf *Rich Clients* bezieht, also weniger auf webbasierte Clients, wurde von der Firma eNode Inc. [eNo02] entwickelt. Mit Hilfe der *eNode UI Markup Language* wird ausgehend von einer WSDL der Aufbau einer Eingabeschnittstelle für den Desktop beschrieben. So wurde beispielsweise ein kleines Mailprogramm entwickelt, welches die E-Mails als SOAP-Nachrichten über einen Web Service zustellt. Als Toolkit wird dabei Java verwendet und die Dialoge mit dem *eNode Object Realizer* aus XML umgesetzt. Es wäre aber auch die Verwendung anderer Toolkits möglich, deren Dia-

[46] Die WSUI-Spezifikation ist auf http://www.wsui.org einsehbar. Die Seite scheint allerdings seit 2002 nicht mehr aktiv zu sein, auch die Beispiele lassen sich nicht einsehen, und die Spezifikation ist noch nicht vollständig.

loge sich aus XML-Daten konstruieren lassen. Das eNode-Konzept sieht nicht vor, dass die Nachrichten direkt aus den Dialogen verschickt werden, sondern es nutzt einen so genannten *Mediator* als Zwischenschicht, der mit Java Beans realisiert wurde (siehe Abbildung 4-15). Der Effekt der Trennung zwischen WSDL und Nutzerschnittstelle ist aber der gleiche.

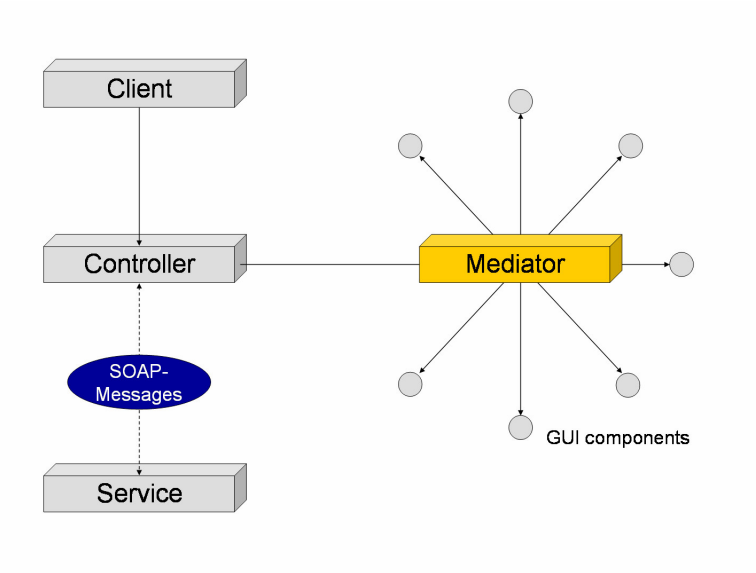

Abbildung 4-15: eNode-Konzept zur Erzeugung einer GUI für Web Services [eNo02]

Unabhängig von WSDL-Spezifikationen gibt es mit SchemoX (*Schema Forms for XML*) [Inf00] einen Versuch, Eingabemasken für XML-Schemata generieren zu lassen. Dabei sollen die Informationen über die einzelnen Elemente aus den zugrunde liegenden XML-Schemata extrahiert und Eingabeformulare in einem generischen Format erzeugt werden. Danach soll dieses generische Format in gängige Formate wie HTML oder Swing[47] umgewandelt werden können. Infozone ist ein Open-Source-Projekt mit dem Ziel, Java- und XML-basierte Komponenten zur Erstellung von komplexen Enterprise Information Portalen (EIP) zu entwickeln. Allerdings ist dieses Projekt im Jahr 2000 das letzte Mal aktualisiert worden.

4.2.6 Eignung im Anwendungskontext und Abgleich mit den gestellten Anforderungen

Die Technologie der Web Services bietet mit SOAP einen mit Hilfe von XML definierten Standard-Kommunikationsmechanismus zwischen verteilten, lose gekoppelten heterogenen Anwendungen und ist damit optimal für die Umsetzung einer SOA im Telearbeitskontext geeignet.

Ein weiterer Vorteil von SOAP gegenüber den bei CORBA, RMI und DCOM eingesetzten Transportprotokollen ist die Art der Datendarstellung. IIOP, ORPC und JRMP sind binäre Protokolle, während SOAP ein textbasiertes Protokoll ist. Als Vorteil stellt sich dar, dass es für den Menschen wesentlich verständlicher ist als ein *binary stream*. Im Gegensatz zu den bei CORBA, RMI und DCOM eingesetzten proprietären und komplexen Protokollen, wurde SOAP von Beginn an so einfach wie möglich gehalten, um die Umsetzung des Protokolls zu erleichtern.

[47] Swing ist eine von Sun Microsystems für die Programmiersprache Java entwickelte Komponentenbibliothek zum Erstellen graphischer Oberflächen.

Mit den Funktionen von UDDI und semantischen Erweiterungen wie OWL-S ist es möglich, Dienste weltweit zu veröffentlichen sowie effizient zu suchen und zu finden. Dies ermöglicht die Nutzung von Diensten externer Service Provider, wodurch sich wirtschaftliche Einsparungspotenziale bei der Integration neuer Anwendungen ergeben können.

Um eine prozessorientierte Abbildung der Arbeitsabläufe und Geschäftsprozesse im Unternehmen zu unterstützen, können die Möglichkeiten standardisierter Kompositionssprachen wie WS-BPEL genutzt werden. Durch die Bereitstellung und Integration komplexer Web Services, deren interne Arbeitsweise und Zusammensetzung dabei weitestgehend transparent bleibt, kann der Aufwand der Integration in die Arbeitsumgebung deutlich verringert werden. Wenn die Komposition bereits auf Provider-Seite erfolgt, können komplexe Anwendungen einfach und schnell integriert werden und dadurch wiederum Einsparungspotenziale erschlossen werden.

4.3 Portaltechnologie

Portaltechnologien haben in den letzten Jahren an Bedeutung gewonnen. Wie bereits in Kapitel 3.3.1 detaillierter beschrieben, haben sich Portale von einfachen Zugangspunkten und Suchmaschinen zu umfangreichen Unternehmensumgebungen entwickelt. Portale bieten eine Vielzahl nützlicher Funktionen, beginnend von der Personalisierung der Benutzeroberfläche und des Funktionsumfangs über Single-Sign-On bis hin zur geschäftsprozessorientierten Integration ganzer Anwendungssysteme. Mit Hilfe eines Portalframeworks lassen sich die verteilten Dienste einer SOA zur einer benutzerzentrierten, prozessbasierten und kollaborativen Anwendung verschmelzen. In den folgenden Kapiteln sollen die technischen Grundlagen der Portaltechnologie näher untersucht werden.

4.3.1 Referenzarchitektur für Portalsoftware

Am Markt gibt es gegenwärtig eine große Anzahl verschiedener Lösungen zur Erstellung von Portalen. Um einen generellen Überblick über den Aufbau und die Funktionsweise von Portalen zu erhalten, hat das Fraunhofer Institut für Arbeitswirtschaft und Organisation in [GuÖ03] eine Referenzarchitektur erstellt, die die notwendigen Bestandteile und angebotenen Funktionalitäten transparent darstellt. Diese Komponenten sind bei den einzelnen Anbietern mehr oder weniger stark ausgeprägt. Abbildung 4-16 zeigt diese in der weiteren Arbeit als Referenz geltende Architektur.

Die Architektur eines Portals ist grundsätzlich in die Bereiche Backend, Anwendungslogik und Präsentation aufgeteilt. Die Benutzerschnittstelle eines Portals ist ein Browser, der über einen Request-Response-Mechanismus (HTTP) mit dem Applikationsserver interagiert, der die Anwendungslogik verwaltet. Zusätzlich zu den Standardfunktionalitäten beinhaltet der Applikationsserver eine Portalengine, die aus Portalbasisdiensten und -anwendungen (sog. *Portlets*) besteht. Diese Portalkomponenten repräsentieren eigenständige, zumeist autonom agierende Anwendungen, welche in einem Container eingebettet sind und über spezielle Interaktionsmechanismen miteinander kommunizieren können. Der Portletcontainer ist eine Laufzeitumgebung für Portlets und für die Kommunikation mit dem Portalframework verantwortlich. Die zentrale Portletengine ist für die Abwicklung sämtlicher Portalinteraktionen verantwortlich. Eingehende HTTP-Anfragen werden zunächst von der Portalengine entgegengenommen und analysiert. Wird mit der Anfrage ein Portlet adressiert, leitet der Server diese an den Portletcontainer weiter. Dieser ermittelt anschließend die für die Ausführung zuständige Portletinstanz.

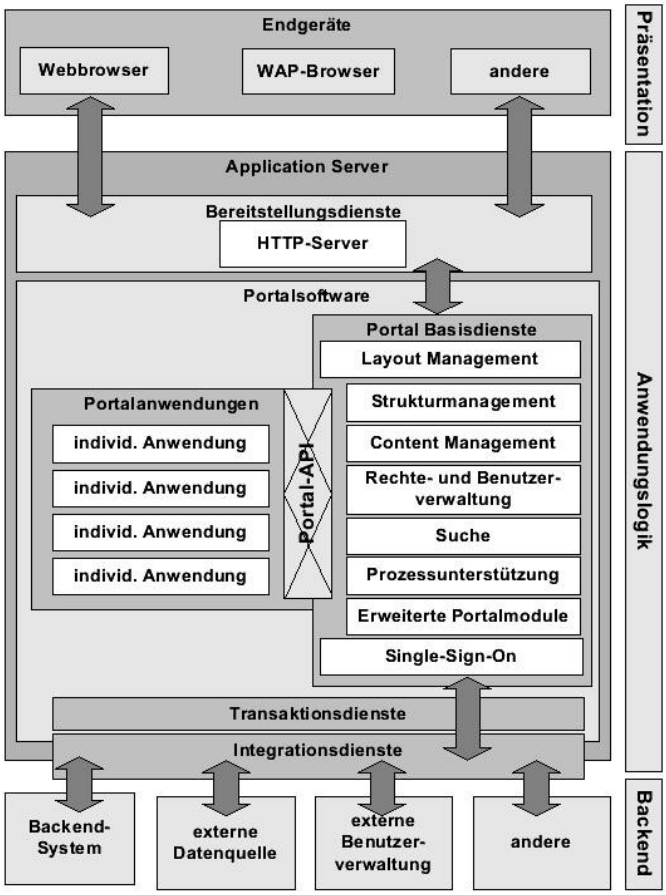

Abbildung 4-16:Referenzarchitektur für Portalsoftware [GuÖ03]

Die Portalbasisdienste (*Shared Services*) stellen die Infrastruktur zur Verfügung, die von den Portlets zur Integration der verschiedenen Anwendungen genutzt wird. Entwickler können sich damit ausschließlich auf die Umsetzung der fachlichen Anforderungen konzentrieren. Ein Beispiel für einen wichtigen Basisdienst ist die Single-Sign-On-Komponente, die von mehreren Portlets für die Authentifikation der Nutzer im jeweiligen Anwendungskontext benötigt wird. Auch die Bereitstellung eines portalweiten Suchdienstes oder eines Content-Management-Systems sind Aufgaben der Portalsoftware. Die Portalbasisdienste interagieren über Transaktions- und Integrationsdienste mit dem Backend-System und externen Anwendungen. Damit die gebotene Funktionalität mit möglichst vielen verschiedenen Endgeräten genutzt werden kann, unterstützen Portale schwerpunktmäßig *markup*-basierte Ausgabemedien, wie bspw. Webbrowser oder WAP-fähige mobile Endgeräte.

4.3.2 Portlets

Als wichtigste Komponenten eines Portals seien hier die Portalanwendungen herausgegriffen. Sie realisieren die eigentliche Anwendungsfunktionalität und werden als Instanz einer allgemeinen Portalanwendungsklasse abgeleitet. Diese Anwendungsklassen sind abhängig vom Portalanbieter und tragen Namen wie *Portlets* oder *I-Views*. Die Portalanwendungen können über die Portal-API aufgerufen werden und über diese auch selbst Portalbasisdienste nutzen.

Portlets sind dynamische, wiederverwendbare Informationskomponenten eines Portals, deren Lebenszyklus von einem Container verwaltet wird. Diese Komponenten konsumie-

ren Anfragen an verschiedenste Anwendungen und Informationsquellen und liefern als Ergebnis dynamische Inhalte (Fragmente) zurück, welche innerhalb des Portalservers zu einer Portalsseite zusammengefasst werden. Ihr Einsatz ermöglicht die Einbettung verschiedenster Anwendungen und Inhalte in die einheitliche Oberfläche eines Portals. Dadurch entsteht eine Arbeitsumgebung, die flexibel und individuell an die Bedürfnisse der Nutzer angepasst und modifiziert werden kann. Portale stellen im Allgemeinen eine Sammlung von Webseiten zur Verfügung, die wiederum eine Anzahl von Portlets enthalten können. So kann das Angebot entsprechend strukturiert und geschachtelt werden. Durch das Portletkonzept können innerhalb kürzester Zeit neue Dienste und Informationen tausenden Portalnutzern zugänglich gemacht werden, ohne dafür die Portalseiten manuell und mit viel Aufwand modifizieren zu müssen.

Die Portalplattformen vieler Hersteller enthalten eine Vielzahl vorgefertigter und voll funktionsfähiger Portlets, die die gewünschte Funktionalität zur Verfügung stellen. Einige dieser Standardkomponenten sind Kollaborationswerkzeuge wie Diskussionsforen und Whiteboards oder informationsverarbeitende Komponenten wie z.B. News-Feeds. Da die meisten Portalanbieter aber bisher bei der Implementierung proprietäre APIs verwenden, ist ein Austausch von Portlets verschiedener Hersteller bisher nicht möglich gewesen. Auch die Entwicklung von Portalkomponenten durch Drittanbieter war deshalb bisher nicht lukrativ, da für jede Portalplattform eine spezielle Version der Komponente entwickelt werden musste.

Zur Standardisierung des Portletkonzeptes wurde im Oktober 2003 von einem Konsortium der wichtigsten Portalhersteller der Standard JSR168 festgeschrieben [JCP03]. Auf der Basis von JSR168 können kompatible Portlets in allen standardkonformen Portalservern genutzt werden. Dies eröffnet Möglichkeiten zum Austausch von Portlets über Herstellergrenzen hinaus und erlaubt erst die Integration von externen Portlets in einem Portal. Nicht Bestandteil der Standardisierung sind portalserverspezifische Funktionalitäten wie das Zusammenfügen der Portlets zu einer Portalseite oder die Platzierung und die individuelle Anpassung der Portlets.

4.3.3 Personalisierung und Individualisierung

Unter Personalisierung oder Individualisierung versteht man im Zusammenhang mit Portalen die Anpassung des Erscheinungsbildes und der Struktur der Portalseiten an die individuellen Bedürfnisse und Anforderungen der einzelnen Nutzer. Dabei stehen dem Anwender eine Reihe von Einstellungsmöglichkeiten zur Verfügung, um die Darstellung der Portlets zu beeinflussen. Neben der Vorgabe von so genannten *Look-and-Feel*-Eigenschaften wie Farben und Formen, ist es dem Nutzer auch möglich, sofern er die nötigen Berechtigungen besitzt, Portlets nach Bedarf einzufügen, zu verschieben oder auch auszublenden. Dazu existieren verschiedene Portletzustände, die durch spezielle Schaltflächen innerhalb der Titelleiste des Portletfensters aktiviert werden können. Die wichtigsten Zustände sind dabei *View* und *Edit*. Im View-Modus werden die Inhaltsfragmente generiert und angezeigt. Im Edit-Modus können verschiedene portletspezifische Einstellungen vorgenommen werden, die für jeden Nutzer individuell gespeichert werden (Individualisierung).

Eine spezielle Funktion der Portlets zur Anpassung des Erscheinungsbildes an die Wünsche der Nutzer ist die so genannte *Decoration*. Dabei kann die Darstellung der Portlets ähnlich zu Fenstern einer Desktop-Anwendung geändert bzw. aus verschiedenen Darstellungsformen ausgewählt werden. Mit den Fensterstatus-Elementen (*Windows State*) kann der Nutzer den Portletstatus aus folgenden Möglichkeiten auswählen:

- Im **Normal-Status** werden mehrere Portlets auf einer Portalseite angezeigt. So erhält der Telearbeiter einen Überblick über alle integrierten Dienste und kann parallel mit

verschiedenen Anwendungen arbeiten. Dabei werden die Portlets in einer optimierten Größe angezeigt. Dieser Status ist als Default-Wert für alle Portlets voreingestellt.

- Der **Maximal-Status** dient dazu, einzelne Portale in maximaler Größe anzuzeigen. Dies kann sinnvoll sein, wenn die Anwendung eine umfangreiche graphische Benutzeroberfläche hat oder viele Informationen auf einer Seite angezeigt werden sollen (z.B. Sortierung einer großen Anzahl von Dokumenten in verschiedene Projekt-Ordner). Alle anderen Portlets werden in diesem Status ausgeblendet.
- Im **Minimal-Status** werden Portlets angezeigt, die momentan nicht genutzt werden müssen oder sowieso nur wenig Platz für die Darstellung benötigen (z.B. einzeilige Maske für die Eingabe von Suchkriterien).

Der Nutzer hat verschiedene Möglichkeiten, die Eigenschaften von Portlets zu beeinflussen. Dabei kann er einerseits den Bearbeitungsmodus der Portalengine nutzen, der eine WYSIWYG-Editierung ermöglicht, oder andererseits direkt die Konfigurationsdateien bearbeiten, die meist in XML beschrieben sind. Die Möglichkeiten der Personalisierung sind von Hersteller zu Hersteller sehr verschieden.

4.3.4 Nutzung von nichtlokalen Portlets

Über WSRP (*Web Services for Remote Portlets*) ist es möglich, ein Proxy-Portlet in ein Portal einzubinden, welches sämtliche Anfragen an ein entfernt befindliches Portlet weiterreicht. Der grundlegende Ablauf ist in Abbildung 4-17 dargestellt: Der „Produzent“ veröffentlicht eine Beschreibung der von ihm angebotenen Portlets (*Service Description Interface*). Der „Konsument“, meist eine Portalanwendung, holt sich die HTML-Beschreibung des Portlets (*Markup Interface*), bietet sie dem Nutzer wie ein eigenes Portlet an und leitet auch alle weiteren Anfragen transparent an das entfernt laufende Portlet weiter.

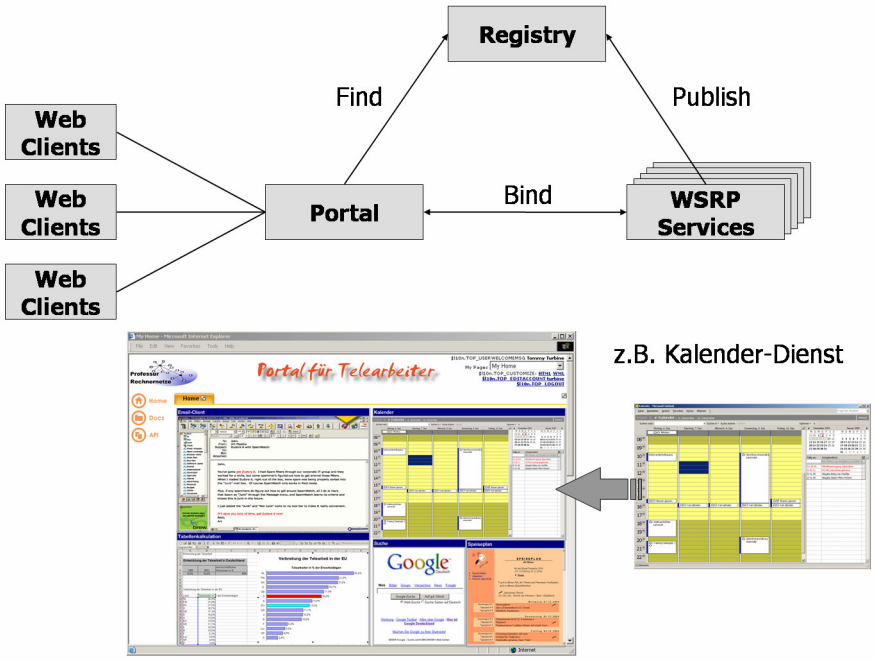

Abbildung 4-17: Integration eines WSRP-Service in das Portal [BrS05]

In WSRP4J ist dies durch ein Proxy-Portlet realisiert. Es werden sowohl „Konsument“ als auch „Produzent“ testweise bereitgestellt. Verfügbar ist WSRP bereits in Portalen wie *Liferay* oder *uPortal*. Diese wurden aus diesem Grund im Rahmen der vorliegenden Arbeit

getestet. *uPortal* verwendet eine Technologie namens *cWebProxy*. Diese arbeitet ähnlich wie WSRP-Markup, indem Webseiten umgeschrieben werden (z.B. Links) und in das Portlet integriert werden. Es gibt auch eine WSRP-Implementierung, die allerdings nicht so einfach mit Beispielen getestet werden kann und nach Meinung eines Testers auch eher unzureichend implementiert ist.

Einige Portale sind bereits WSRP-fähig, leider ist das bei JetSpeed 1.6 noch nicht der Fall. Da JetSpeed als Portletcontainer Pluto verwendet, muss man abwarten, bis Pluto die Bibliothek WSRP4J vollständig integriert hat. Leider scheint WSRP4J auch nicht mehr aktiv entwickelt zu werden. In *Liferay* ist tatsächlich ein WSRP-Proxy-Portlet integriert. Dieses benötigt aber einen „Produzenten“ und greift standardmäßig auf die Oracle-Portalseite zu, auf der es sowohl Testportlets gibt, als auch die Möglichkeit, eigene Portlets zu exportieren und die Funktionsweise zu überprüfen. Leider ist es nicht möglich, das WSRP-Proxy-Portlet von Liferay direkt in JetSpeed zu verwenden, da zumindest die Anbindung an Velocity anders gelöst zu sein scheint. Wenn man ein bereits für JetSpeed erstelltes Portlet in Liferay verwenden möchte, so sollte es in die Standardform (Format WAR) gebracht werden, und kann dann über *Hot-Deployment* via *Ant* automatisch registriert und hinzugefügt werden.

4.3.5 Eignung im Anwendungskontext und Abgleich mit den gestellten Anforderungen

Portale bieten einen einheitlichen Zugangspunkt zu den verschiedensten Anwendungen. Sie unterstützen die Anpassung der Arbeitsumgebung an die Bedürfnisse der einzelnen Nutzer (Personalisierbarkeit) und an verschiedene Endgeräte (Adaptierbarkeit). Weiterhin bieten sie wichtige Sicherheitsfunktionen wie Authentifizierung und Single-Sign-On an.

Mit der Darstellung im Browser wird die Anforderung an eine webbasierte Thin-Client-Lösung erfüllt. Der Telearbeiter kann von jedem beliebigen Endgerät, das über einen Internetzugang und einen Browser oder äquivalente Client-Programme verfügt, auf das Portal zugreifen. Der Installationsaufwand auf der Seite des Telearbeitsplatzes kann somit deutlich verringert werden.

4.4 Fazit

Auf der Basis von Web Services lässt sich eine Telearbeitsumgebung schaffen, die im Vergleich mit vorhandenen Ansätzen universeller einsetzbar ist. Ziel ist die automatische Auswahl, Verbindung und Interoperation von geeigneten Web Services, um alle Aufgaben des Telearbeiters zu lösen.

Bisherige Kommunikationsarchitekturen wie CORBA, RMI und DCOM können diese Anforderungen nur teilweise erfüllen. Sie bieten nicht die Flexibilität, die für eine unkomplizierte Umsetzung von Transaktionen und Dienstleistungen bei der wechselnden Interaktion von lose gekoppelten Internetanwendungen benötigt werden [Coy02]. Das Konzept der Web Services versucht hier eine Lösung anzubieten und stellt eine Reihe von "offenen Standards" zur Verfügung, die unabhängig von Plattform, Betriebssystem und Programmiersprache eingesetzt werden können [W3C02a]. Die Web-Service-Technologie mit ihren Möglichkeiten für den standardisierten Nachrichtenaustausch zwischen B2B-Partnern wird früher oder später auch innerhalb der Unternehmen zu einer Konkurrenz der proprietären Kommunikationsarchitekturen wie CORBA, RMI und DCOM werden.

Mit Hilfe von standardisierten Kompositionssprachen wie WS-BPEL können Geschäftsprozesse und Arbeitsabläufe auf komplexe Web Services transparent abgebildet werden. Die Definition von *Abstract Processes* und die semantische Beschreibung mit OWL-S

erlauben die flexible dynamische Auswahl und Integration verschiedenster Dienste in komplexen Workflows.

Um die unterschiedlichen Dienste für den Benutzer in einer einheitlichen Oberfläche zusammenzufassen, eignet sich die Portaltechnologie. Sie bietet neben der flexiblen Integration verschiedenster Anwendungen als Portlets auch Werkzeuge zur individuellen Anpassung der Oberfläche an die Bedürfnisse der Nutzer (Personalisierbarkeit) und an verschiedene Endgeräte (Adaptierbarkeit). Zusätzlich stellen heutige Portallösungen umfangreiche Sicherheitswerkzeuge, wie z.B. zur Verwaltung hierarchischer Rollen- und Nutzerkonzepte und zum *Single-Sign-On*, zur Verfügung.

Aus den genannten Gründen wird für die Implementierung der universellen Telearbeitsumgebung die Nutzung der Web-Service-Technologie zur Realisierung der serviceorientierten Architektur und die Nutzung der Portaltechnologie zur benutzerorientierten Integration vorgeschlagen.

KAPITEL 5
FACHLICHES UND TECHNOLOGISCHES KONZEPT EINER UNIVERSELLEN TELEARBEITSUMGEBUNG

"Es gibt keine ‚schlüsselfertigen' Lösungen. Sollen Technologien wirklich flexibel sein, müssen sie in die soziale Organisation des Arbeitsplatzes integriert werden, damit eine wettbewerbsfähige Kombination von Produktivität, Leistung und Qualität erreicht werden kann."

Europäische Kommission; Grünbuch Nr. 28, 1996

Dieses Kapitel stellt das fachliche und technologische Konzept zur Entwicklung einer universellen Telearbeitsumgebung auf Basis einer serviceorientierten Architektur detailliert vor und bildet somit den Kern dieser Dissertation. Dazu wird zu Beginn die Architektur des Gesamtsystems beschrieben und die Auswahl der Web-Service-Technologie zu deren Umsetzung begründet. Anschließend werden die einzelnen Komponenten und Funktionen des Systems näher spezifiziert und Möglichkeiten zu deren praktischer Umsetzung diskutiert.

5.1 Architektur einer universellen Telearbeitsumgebung

Aufbauend auf die in Kapitel 2.6 definierten Anforderungen wird nun eine Architektur der Telearbeitsanwendung entworfen und beschrieben. Wichtiges Kriterium für die Flexibilität der Telearbeitsumgebung ist die Modularität der angestrebten Lösung. Dabei sollen wiederverwendbare Komponenten in einer Bausteinarchitektur zur Verfügung gestellt werden, um die Telearbeitsumgebung flexibel konfigurieren und an die Bedürfnisse eines jeden Telearbeiters anpassen zu können. Um dies zu erreichen wurden die Paradigmen der *Service Oriented Architecture* (SOA) – also der Verwendung verteilter, lose gekoppelter Dienste – zugrunde gelegt. So soll der Telearbeiter die Möglichkeit haben, aus einem Set verschiedener Telearbeitsdienste die für seine Arbeit benötigten Dienste auszusuchen und diese flexibel in seine Arbeitsumgebung zu integrieren. In den folgenden Kapiteln wird die benötigte Infrastruktur vorgestellt, um eine solche dienstorientierte Architektur abzubilden.

5.1.1 Serviceorientierte Architektur der Telearbeitsumgebung

Wie die Paradigmen der SOA nun in die Architektur der Telearbeitsumgebung integriert werden können, wird im folgenden Abschnitt beschrieben.

Ein zentraler *Teleworking Service Integrator* fügt sich dabei zwischen *Service Consumer* und *Service Provider* ein (siehe Abbildung 5-1) und übernimmt dabei die Funktion eines *Service Brokers*, indem er sowohl vorhandene Telearbeitsdienste registriert und verwaltet als auch weitere benötigte Basisdienste sucht.

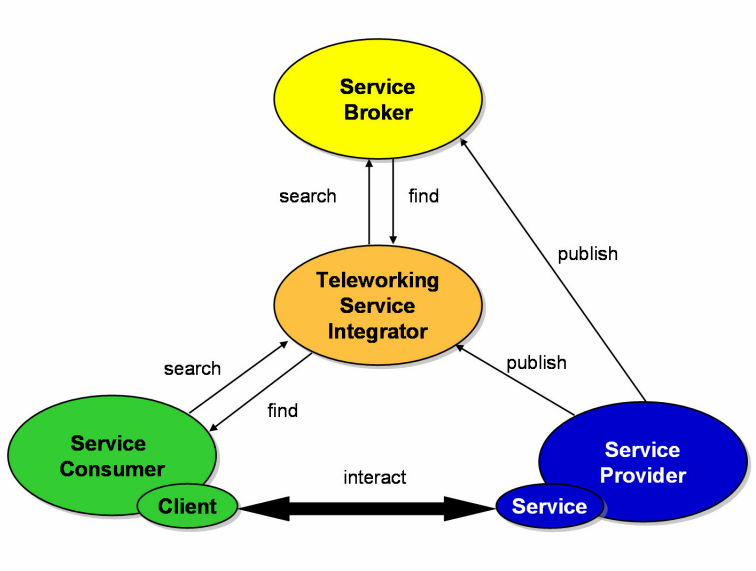

Abbildung 5-1: Einordnung des *Teleworking Service Integrator* in eine serviceorientierte Architektur

Der *Teleworking Service Integrator* hat weiterhin die Aufgabe, die gefundenen Telearbeitsbasisdienste zu neuen komplexen Diensten zu orchestrieren, um komplexe Geschäftsprozesse und Arbeitsabläufe abbilden zu können. Diese komplexen Telearbeitsdienste stellt er dann dem *Consumer*, also dem Telearbeiter, zur Verfügung, der sie dann in seine Arbeitsumgebung integrieren kann (siehe Abbildung 5-2).

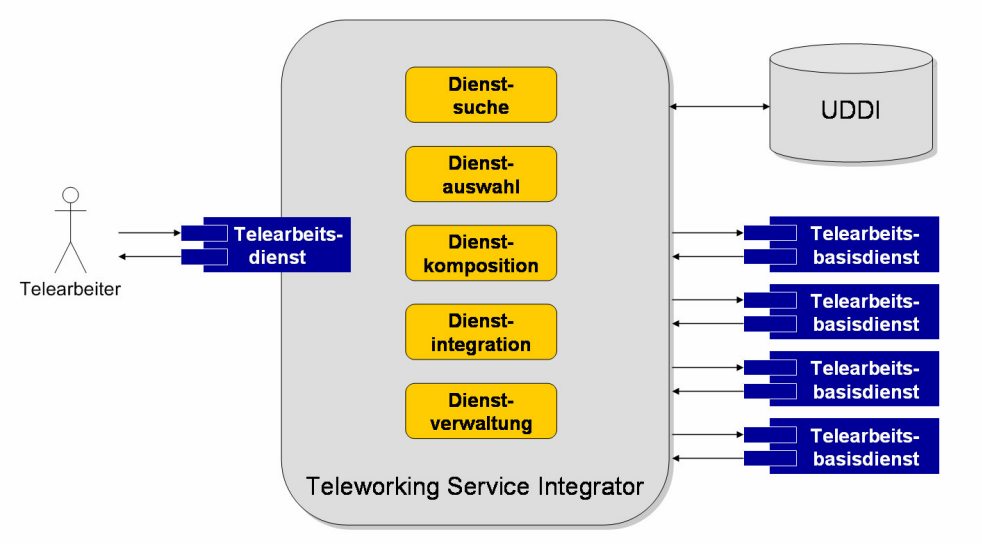

Abbildung 5-2: Funktionsweise des *Teleworking Service Integrator*

Die benötigten Telearbeitsdienste werden aus unterschiedlichen Basisdiensten komponiert, die von verschiedenen Anwendungssystemen innerhalb des Unternehmens oder externen Service Providern angeboten werden. Die einzelnen Dienste können selbst auch komplexe Strukturen enthalten. Diese Komplexität wird aber durch die klaren Schnittstellen der Dienste gekapselt und bleibt somit „Geheimnis“ des Service Providers. Diese einfache Architektur schafft Flexibilität. Einzelne Dienste können ersetzt oder neu komponiert werden, ohne dass der Gesamtprozess geändert werden muss. Durch die Bildung von wiederverwendbaren Diensten aus Anwendungen und Informationsobjekten können die Komponenten der Telearbeitsumgebung modular zusammengestellt und daher auch schnell verändert werden. Somit können die bereitgestellten Inhalte, Dienste und Funktionen beliebig konfiguriert und zudem nutzerspezifisch angepasst werden.

Um die Nutzer bei der Suche und Integration geeigneter Dienste zu unterstützen, soll im Rahmen dieser Arbeit ein Framework geschaffen werden, das die prozessorientierte Komposition komplexer Dienste aus einem Pool von wiederverwendbaren Basisdiensten mit Hilfe semantischer Beschreibungen vereinfacht. Um alle benötigten Dienste und Anwendungen in einer gemeinsamen Arbeitsumgebung vereinen zu können, ist letztendlich noch eine benutzerorientierte Integration notwendig, für die ebenfalls ein Lösungsansatz vorgestellt wird.

5.1.2 4-Schichten-Architektur

Der optimale Nutzen einer serviceorientierten Architektur lässt sich nur erreichen, wenn deren Einführung sich nicht nur auf die technische Implementierung einzelner Anwendungen beschränkt, sondern sich an den Prozessen der Anwender orientiert. So wird in der Implementierung bisher meist ein Bottom-Up-Ansatz gewählt, bei dem die Entwicklung ausgehend von den vorhandenen Systemen schrittweise von unten nach oben bis zu den Benutzerschnittstellen, also bis zum Anwender hin, durchgeführt wird. Sinnvoller ist es aber, wenn der Entwurfsprozess bei den Benutzeroberflächen und Prozessen beginnt, um dann die Anforderungen an die Endsysteme zu definieren.

Deshalb wurde im Rahmen dieser Arbeit für die Realisierung einer flexiblen Telearbeitsumgebung im Gegensatz zu herkömmlichen 3-Schichten-Architekturen von Webanwendungen eine 4-Schichten-Architektur (*4-Tier Architecture*) gewählt (siehe Abbildung 5-3). Neben den bekannten Daten- und Anwendungsschichten werden zwei neue Schichten eingefügt, da die Integration der verschiedenen Dienste auf zwei Ebenen stattfindet - auf der Ebene der Benutzeroberflächen und der Ebene der Prozesse. Aufbauend auf die Anwendungsschicht wird eine Schicht zur prozessorientierten Integration aufgesetzt, die die Komposition der in der Anwendungsschicht durch verschiedene Provider bereitgestellten Basisdienste zu komplexen Telearbeitsdiensten realisiert. Erst in der übergeordneten Schicht wird analog zur Präsentationsschicht in herkömmlichen 3-Tier-Architekturen die benutzerorientierte Integration der verschiedenen Anwendungen realisiert.

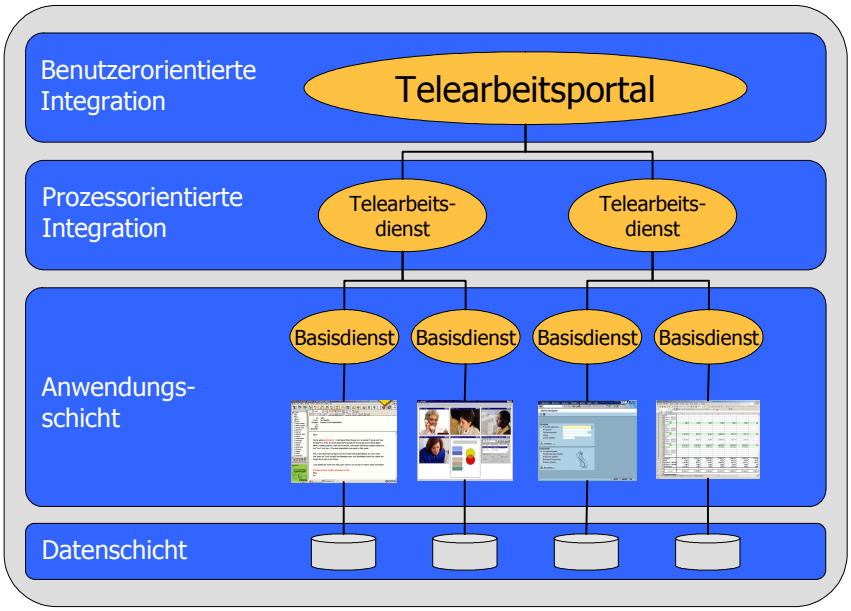

Abbildung 5-3: 4-Schichten-Architektur der Telearbeitsumgebung [BrS05]

Diese 4-Schichten-Architektur ermöglicht die Trennung der Anwendungen auf der Seite des Dienstanbieters (*Service Provider*) und der Integration sowie Darstellung auf der Seite des Dienstnutzers (*Service Consumer*). Dabei ist es aber auch denkbar, dass schon auf der Seite des Providers eine Prozessintegration stattfindet und komplexe Dienste zur Verfügung gestellt werden.

5.1.3 Aktoren des Systems

Zunächst ist es wichtig zu analysieren, welche Aktoren beteiligt sind und wie sie das System benutzen. Aktoren definieren verschiedene Rollen von Nutzern des Systems. Grundsätzlich sind in der vorgeschlagenen Architektur folgende Rollen zu unterscheiden:

Der **Service Provider** stellt einen Dienst als Web Service zur Verfügung, indem er die WSDL-Beschreibung des Dienstes und seine Koordinaten in einem UDDI-Registry veröffentlicht. *Service Provider* beschreibt allgemein einen Dienstanbieter, also ein Unternehmen, das Dienstleistungen als Web Services zur Verfügung stellt. Dies kann im konkreten Anwendungsfall auch das eigene Unternehmen des Telearbeiters sein, das spezielle Kommunikations- und Kollaborationsdienste für seine Mitarbeiter oder die Schnittstellen zu unternehmensinternen Anwendungen als Web Services anbietet. Dazu ist es ausreichend, dass die Beschreibung der Web Services und ihrer Schnittstellen in einem nichtöffentlichen UDDI-Register erfolgt.

Der **Portal-Admin** oder **Entwickler**, der in den meisten Fällen ein Mitarbeiter der IT-Abteilung des Unternehmens des Telearbeiters sein wird, entwirft und komponiert komplexe Dienste, die die Geschäftsprozesse des Unternehmens und Arbeitsabläufe des Telearbeiters in WS-BPEL abbilden. Dabei wird ihm ein Framework bereitgestellt, mit dem die Integration und Wiederverwendung von Basisdiensten vereinfacht werden kann. Bei der Suche benötigter Dienste kann er aus einem Satz vorhandener Basisdienste auswählen, die somit wiederverwendet werden können, oder eine semantik-unterstützte Suche in einer öffentlichen UDDI-Registry durchführen. Bei der Komposition greift er direkt auf die WSDL der verschiedenen Web Services zu. Zur Integration der entwickelten Dienste in das Portal nutzt er die Funktionalitäten der Portalengine. Er ist mit der Web-Service- und Portaltechnologie und den dazugehörigen Spezifikationen, Protokollen und Werkzeugen vertraut.

Die dritte Rolle beschreibt den eigentlichen Endnutzer des Systems, also den einzelnen **Telearbeiter**. Er will verschiedene Dienste flexibel und schnell in seine Arbeitsumgebung integrieren, ohne dabei spezielle Kenntnisse von der zugrunde liegenden Technologie und den verwendeten Werkzeugen zu besitzen. Deshalb muss ihm ein Framework zur Verfügung gestellt werden, in dem er Suchanfragen einfach und intuitiv generieren kann und die Integration der Dienste in das Portal automatisiert wird.

Wie die notwendigen Funktionalitäten zur Unterstützung der einzelnen Rollen implementiert werden können, wird in den folgenden Kapiteln beschrieben.

5.1.4 Abbildung der Architektur auf Web-Service-Technologie

Die vorgeschlagene serviceorientierte Architektur lässt sich am besten unter Nutzung der Web-Service-Technologie und ihrer zugehörigen Protokolle zur Dienstbereitstellung, Komposition und Integration und unter Nutzung der Portaltechnologie für die benutzerorientierte Integration implementieren.

Die benötigten Basisdienste werden als Web Services implementiert, die Schnittstellen mit WSDL beschrieben und vom Service Provider in einem UDDI-Register veröffentlicht. Dabei können die Web Services auch Schnittstellen zu vorhandenen Legacy-Anwendun-

gen im Unternehmen zur Verfügung stellen. Komplexere Abläufe können mit Hilfe der Web-Service-Kompositionssprachen wie WS-BPEL prozessorientiert abgebildet und aus einfachen Web Services komponiert werden. Dienste können über UDDI-Suchanfragen gefunden und dann in eine Weboberfläche integriert werden. Zur effektiveren Suche ist es sinnvoll, semantische Beschreibungen der Dienste z.B. unter Verwendung von RDF oder OWL-S zu benutzen. Die nutzbaren Technologien wurden bereits in Kapitel 4 näher vorgestellt.

Als Frontend für die heterogene Anwendungslandschaft werden Portalframeworks eingesetzt, da sie die Paradigmen der serviceorientierten Architektur unterstützen und umsetzen. Sie verbinden die einzelnen Anwendungen zu einer für den Nutzer zusammengehörigen Arbeitsumgebung. Die Nutzung der Portaltechnologie ermöglicht außerdem die Personalisierung und Individualisierung der Arbeitsumgebung entsprechend den Bedürfnissen und Wünschen jedes Telearbeiters. Portale bieten umfangreiche Funktionen zur Verwaltung von Nutzern und Rollen und deren zugeordneten Profilen und Rechten. Mittels Single-Sign-On ermöglichen sie einen sicheren Zugang zu allen benötigten Diensten und Anwendungen. Portale unterstützen bereits die Anpassung der Benutzeroberfläche an verschiedene Endgeräte und erfüllen damit auch die Forderung nach Adaptierbarkeit der Lösung.

Bisher ungeklärt ist die Frage, wie die Integration der Web Services in das Portal erfolgen kann. Sie soll automatisiert erfolgen, damit auch dem „unbedarften“ Nutzer die flexible Integration neuer Dienste ermöglicht werden kann. Hier sucht die vorliegende Arbeit nach neuen Ansätzen zur Beschreibung von Benutzerschnittstellen für Web Services, da diese originär für die Kommunikation von Anwendungen untereinander in Form von *Remote Procedure Calls* gedacht waren und nicht für die direkte Nutzung durch Anwender. Deshalb bieten Web Services auch selten eigene Benutzerschnittstellen an. Die Entwicklung geht aber weiter zu komplexen Diensten und Dienstleistungen, die mit Hilfe von Web Services erbracht werden, und damit auch zur Loslösung vom RPC-Paradigma hin zu ganzheitlichen Lösungen.

5.2 Modellierung, Klassifizierung und Realisierung der benötigten Basisdienste

Grundlage der Modellierung der Telearbeitsdienste ist die Abbildung realer Prozesse und Abläufe im Büro auf verschiedene webbasierte Dienste. Komplexere Prozesse und Abläufe sollten dabei in einfache Dienste aufgeteilt und als Web Services implementiert werden. Ein Service modelliert dabei ein in sich abgeschlossenes Stück Geschäftslogik oder eine bestimmte Funktionalität. Häufiger Diskussionspunkt ist dabei die Granularität eines Dienstes. Potenzial für Wiederverwendung entsteht aber nur, wenn die Dienste einerseits genügend grob granular sind, um eine in sich geschlossene Funktionalität anzubieten. Andererseits müssen sie jedoch auch fein granular genug sein, um in unterschiedlichen Kontexten Verwendung finden zu können. Dafür gibt es aber keine allgemeingültigen Kriterien.

Um den Entwickler bei der Modellierung zu unterstützen, soll im Rahmen dieser Arbeit ein Framework entwickelt werden, dass die Komposition von komplexen Diensten aus wiederverwendbaren Basisdiensten unterstützt. Dazu gilt es zunächst zu analysieren, welche Telearbeitsdienste angeboten werden sollen und wie sie modelliert und beschrieben werden können. Dazu wird in den folgenden Abschnitten eine Klassifizierung der benötigten Basisdienste vorgenommen und deren Modellierung als Web Services beschrieben. Um auch auf vorhandene Legacy-Anwendungen im Unternehmen zugreifen zu können, sollen durch Web Services Schnittstellen zu diesen Systemen zur Verfügung gestellt werden.

Aus der Analyse der für Telearbeit relevanten Anforderungen an eine universelle Softwarelösung ergeben sich folgende Kategorien notwendiger Basisdienste, die eine Telearbeitsumgebung zur Verfügung stellen sollte. Diese wurden anhand des Anwendungskontextes und der bereitgestellten Funktionalität klassifiziert.

Anwendungsdienste ermöglichen den Zugriff auf alle benötigten Anwendungen und Dokumente, z.B. Office-Anwendungen, Datenbankzugriffe oder betriebswirtschaftliche Software.

Kommunikationsdienste dienen der Bereitstellung aller grundlegenden Funktionalitäten für die Kommunikation. Dabei sollen sowohl synchrone (Videokonferenzen, Internet-Telefonie, Chat) als auch asynchrone Kommunikationsformen (E-Mail, News, Diskussionsforen, Fax) unterstützt werden.

Kollaborationsdienste gewährleisten die Zusammenarbeit aller an einer Aufgabe beteiligten Personen, indem sie Werkzeuge für Dokumentenaustausch und -verwaltung oder Informations- und Wissensmanagement zur Verfügung stellen.

Koordinationsdienste koordinieren die Zusammenarbeit mehrerer Personen, die gemeinsam eine Aufgabe erfüllen müssen. Dazu gehören die Verteilung der Arbeitsaufgaben und Projekte, Terminplanung und -verwaltung sowie die Verwaltung, Abrechnung und Sicherung der erledigten Aufgaben.

Sicherheitsdienste ermöglichen die Bereitstellung von Sicherheitsfunktionalität auf allen Ebenen, z.B. zur Benutzerverwaltung, Authentifizierung und Sicherung der Datenübertragung. Dazu gehört auch ein Monitoring in allen Schichten (z.B. der Dienst- und Dokumentenzugriffe).

Die einzelnen Kategorien werden in den folgenden Kapiteln detaillierter erläutert.

5.2.1 Anwendungsdienste

Über die Anwendungsdienste soll der Telearbeiter Zugriff auf alle Anwendungen erhalten, die er für die Erledigung seiner Aufgaben benötigt. Diese Dienste sollen Schnittstellen zu vorhandenen Anwendungen im Unternehmen wie z.B. Office-Anwendungen, Datenbankzugriffen oder betriebswirtschaftlicher Software herstellen. In den meisten Unternehmen übernehmen nach wie vor diese so genannten Legacy-Systeme alle dort anfallenden Aufgaben von der Buchhaltung bis hin zur Vertriebsautomatisierung. Deshalb ist die Einbindung dieser Anwendungen über standardisierte Schnittstellen so wichtig.

Grundvoraussetzung für die Integration ist aber, dass die Anwendungen eine Web-Service-Schnittstelle zur Verfügung stellen müssen. Diese Hürde wird im Laufe der Zeit in dem Maße an Bedeutung verlieren, in dem sich die Einhaltung der Paradigmen der SOA bei der Systementwicklung weiter durchsetzen kann und die Betriebs- und Anwendungsplattformen einen immer größeren Teil ihrer Ressourcen über Web Services bereitstellen. Zurzeit bieten jedoch nur wenige Legacy-Plattformen Web-Service-Schnittstellen an. Es ist aber möglich, diese Schnittstellen über einen Adapter bereitzustellen und damit bestehende Anwendungen nachträglich Web-Service-tauglich zu machen. Dieses so genannte *Web Service Gateway* setzt auf vorhandene API's der Legacy-Anwendung auf und wandelt alle eingehenden SOAP-Nachrichten in entsprechende Funktionsaufrufe der Anwendung um und alle Ausgaben wiederum in SOAP-Nachrichten (siehe Abbildung 5-4). Damit ein SOAP-Client die Anwendung auch nutzen kann, muss das Gateway auch eine Schnittstellenbeschreibung in WSDL zur Verfügung stellen.

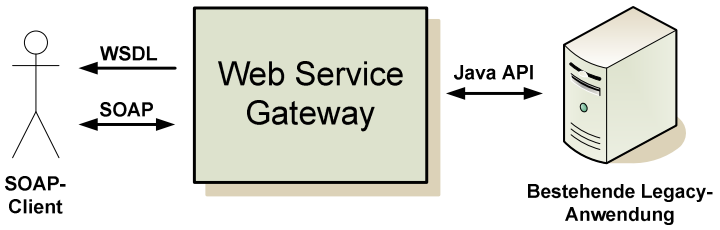

Abbildung 5-4: Funktionsweise eines Web Service Gateway

Zu den wichtigsten Anwendungen, die in eine Telearbeitsumgebung integriert werden sollten, zählt die ganze Palette der Office-Anwendungen wie Textverarbeitung, Tabellenkalkulation und Präsentationserstellung, die in heutigen Büroumgebungen nicht mehr wegzudenken sind. Ein weiterer großer Anwendungsbereich sind die betriebswirtschaftlichen Anwendungen (*Business Software*) zur Unterstützung von *Customer Relationship Management* (CRM), *Enterprise Resource Planning* (ERP), *Human Resources Management* (HRM) oder *Supply Chain Management* (SCM), um nur einige zu nennen.

Auch Werkzeuge zum Design von Webseiten, Programmierumgebungen und Entwicklungswerkzeuge sollen über die Telearbeitsumgebung zur Verfügung gestellt werden. Weiterhin sollten Grafikbearbeitungsprogramme und CAD-Systeme nicht vergessen werden, obwohl sich deren Integration in eine Webumgebung etwas schwieriger gestaltet, da sie eine sehr umfangreiche Benutzeroberfläche benötigen.

5.2.1.1 Dienst zum Bereitstellen von Kundendaten

Mit diesem Anwendungsdienst soll ein Zugriff auf das *Customer Relationship Management System* (CRMS) des Unternehmens ermöglicht werden, um sich zu einem Kunden die vorhandenen Kundendaten anzeigen zu lassen. Heutige CRM-Systeme werden in der Regel auf Basis einer Datenbank realisiert, mit deren Hilfe eine strukturierte und ggf. automatisierte Erfassung sämtlicher Kundenkontakte und -daten ermöglicht wird. Zur automatischen Auswertung der Daten werden Verfahren wie *Data Mining* oder OLAP (*Online Analytical Processing*) zur Verfügung gestellt. Auf Grundlage dieser Kundendatenbank gibt es eine Vielzahl von Kunden- und Marketingprozessen, die durch entsprechende CRM-Anwendungen unterstützt werden. Mit einer entsprechenden Web-Service-Schnittstelle können verschiedene Basisdienste geschaffen werden, die in vielen Arbeitsabläufen und Geschäftsprozessen wiederverwendet werden können.

Um die Kundendaten bereitstellen zu können, muss der Web Service eine Operation anbieten, die bei Eingabe eines Kundennamens aus dem CRM-System die passenden Kundendaten filtert und ausgibt. Diese Operation besitzt ein Interface, welches in diesem Fall dem Request-Response-Pattern entsprechen muss, da auf eine Anforderung an den Web Service ein Ergebnis zurückgeliefert werden soll. Als Input-Parameter wird der Name des Kunden übergeben, als Ausgabe sollen Kundendaten wie Anschrift, Homepage, E-Mail-Adresse und Telefonnummer zurückgegeben werden. Die WSDL-Dienstbeschreibung ist in Listing 5-1 auszugsweise dargestellt.

```
<definitions ...>
  <types>
    <s:schema elementFormDefault="qualified"
     targetNamespace="http://www.telework-inc.de/KundendatenBereitstellen/">
      <s:element name="kundendatenAbfragenResponse">
        <s:complexType>
          <s:sequence>
             <s:element minOccurs="0" maxOccurs="1" name="name" type="s:string" />
             <s:element minOccurs="0" maxOccurs="1" name="homepage" type="s:string" />
             <s:element minOccurs="0" maxOccurs="1" name="email" type="s:string" />
             <s:element minOccurs="0" maxOccurs="1" name="ort" type="s:string" />
             <s:element minOccurs="0" maxOccurs="1" name="strasse" type="s:string" />
```

```
            <s:element minOccurs="0" maxOccurs="1" name="telefon" type="s:string" />
          </s:sequence>
        </s:complexType>
      </s:element>
    </s:schema>
  </types>
  <message name="kundendatenAbfragenSoapIn">
      <part name="kundenname" type="s:string" />
  </message>
  <message name="kundendatenAbfragenSoapOut">
      <part name="kundendaten" element="s0:kundendatenAbfragenResponse" />
  </message>
  <interface name="CRMAnfrageSOAP">
    <operation name="kundendatenAbfragen"
     pattern="http://www.w3.org/2003/06/wsdl/request-response">
      <input message="s0:kundendatenAbfragenSoapIn" />
      <output message="s0:kundendatenAbfragenSoapOut" />
    </operation>
  </interface>
</definitions>
```

Listing 5-1: Interface-Beschreibung in der WSDL des Web Service `KundendatenBereitstellen`

Weitere denkbare Ausgabeparameter wären z.B. eine Liste von Web Services, die der Kunde zur Zusammenarbeit anbietet, oder ein öffentlicher Schlüssel des Kunden, mit dem man Nachrichten an ihn verschlüsseln kann, oder die Zugangsdaten für das Portal oder den Zugriff auf spezielle Dienste des Kunden. Diese Daten müssen dazu natürlich im CRM-System des eigenen Unternehmens abgelegt und verwaltet werden.

Dieser Anwendungsdienst kann nun in verschiedene komplexe Abläufe und Geschäftsprozesse integriert werden, die die Kundendaten zur Weiterverarbeitung nutzen (siehe auch Kapitel 5.3.3).

5.2.2 Kommunikationsdienste

Besonders wichtig für Telearbeiter ist die Bereitstellung aller grundlegenden Funktionalitäten für die Kommunikation mit Kollegen und Vorgesetzten. Ziel des Einsatzes solcher Dienste ist es, Reaktionszeiten auf Anfragen zu minimieren, Probleme bei der Teamarbeit zu beseitigen und den Dialog zwischen den Mitarbeitern zu fördern, also zusammengefasst, die interne Unternehmenskommunikation zu verbessern.

Dabei sollen sowohl synchrone als auch asynchrone Kommunikationsformen unterstützt werden. Beispiele für **synchrone** Kommunikationsdienste sind:

- Videokonferenzen,
- Web-Conferencing,
- Internet-Telefonie (VoIP),
- Instant Messaging,
- Chat.

Zur Unterstützung **asynchroner** Kommunikation wären folgende Dienste denkbar:

- E-Mail,
- News,
- Diskussionsforen,
- Fax,
- SMS.

5.2.2.1 E-Mail-Dienst

Ein Beispiel für einen solchen Kommunikationsdienst wäre der Zugriff auf das E-Mail-Postfach des Telearbeiters mittels eines Web Service. Vorteil der Implementierung als Web Service ist, dass man die Methoden des Dienstes auch von anderen Clients, z.B. mobilen Endgeräten, aufrufen kann. So kann sich ein mobiler Telearbeiter bspw. seine E-Mails auf das Handy oder ein PDA laden, während er unterwegs zu einem Kunden ist. Die Client-Oberfläche wird dann entsprechend des Endgerätes adaptiert und auch die aufgerufenen Methoden können daran angepasst werden, indem z.B. die Rückgabeparameter eingeschränkt werden.

Der E-Mail-Dienst sollte dabei folgende Funktionen anbieten, wobei nicht alle zwingend umgesetzt werden müssen:

- Empfangen von E-Mails via POP3 und IMAP,
- Versenden von E-Mails via SMTP,
- Anhängen von Dateien an die E-Mail (Attachments),
- Verwaltung von E-Mails:
  - Lesen/Schreiben/Speichern/Löschen von E-Mails,
  - Anlegen von Ordnern,
  - Sortieren der E-Mails in verschiedenen Ordnern,
  - Antworten auf E-Mails mit Einfügetext (Reply).
- Filtern von E-Mails,
- Adressverwaltung:
  - Editieren/Speichern/Löschen von Adressen,
  - Definition von Adress-Aliasen.
- zusätzliche Funktionen:
  - Autoanswering bei Abwesenheit etc.,
  - Einstellen mehrerer POP3-, IMAP- und SMTP-Konten,
  - Wahl eines SMTP-Servers zum Senden,
  - E-Mail-Ticker (Hinweis, wenn neue E-Mails eintreffen).

Die einzelnen Funktionen können nun als Methoden eines Web Service implementiert und die Schnittstellen dafür in WSDL beschrieben werden. Als vereinfachtes Beispiel soll hier der Web Service `emailService` mit der Methode `SendEmail` herausgegriffen werden, dessen WSDL in Listing 5-2 dargestellt ist.

```
<?xml version="1.0" encoding="utf-8" ?>
<wsdl:definitions xmlns:s="http://www.w3.org/2001/XMLSchema"
 xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
 xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
 xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
 xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/"
 xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
 xmlns:tns="http://rn.inf.tu-dresden.de/xmlservices/Email"
 targetNamespace="http://rn.inf.tu-dresden.de/xmlservices/Email">
  <wsdl:types>
    <s:schema elementFormDefault="qualified"
     targetNamespace="http://rn.inf.tu-dresden.de/xmlservices/Email">
      <s:element name="EmailResponse">
        <s:complexType>
          <s:sequence>
            <s:element minOccurs="0" maxOccurs="1" name="resultcode" type="s:string" />
          </s:sequence>
        </s:complexType>
      </s:element>
```

```
      <s:element name="EmailRequest">
        <s:complexType>
          <s:sequence>
            <s:element minOccurs="0" maxOccurs="1" name="message" type="tns:MailMessage"
             />
          </s:sequence>
        </s:complexType>
      </s:element>
      <s:complexType name="MailMessage">
        <s:sequence>
          <s:element minOccurs="1" maxOccurs="1" name="from" type="tns:string" />
          <s:element minOccurs="1" maxOccurs="1" name="to" type="tns:string" />
          <s:element minOccurs="1" maxOccurs="1" name="subject" type="s:string" />
          <s:element minOccurs="1" maxOccurs="1" name="message" type="s:string" />
        </s:sequence>
      </s:complexType>
    </s:schema>
  </wsdl:types>
  <wsdl:message name="EmailSoapIn">
    <wsdl:part name="parameters" element="tns:EmailRequest" />
  </wsdl:message>
  <wsdl:message name="EmailSoapOut">
    <wsdl:part name="parameters" element="tns:EmailResponse" />
  </wsdl:message>
  <wsdl:portType name="EmailSoap">
    <wsdl:operation name="Email">
      <documentation xmlns="http://schemas.xmlsoap.org/wsdl/">Send an email.
      </documentation>
      <wsdl:input message="tns:EmailSoapIn" />
      <wsdl:output message="tns:EmailSoapOut" />
    </wsdl:operation>
  </wsdl:portType>
  <wsdl:binding name="EmailSoap" type="tns:EmailSoap">
    <soap:binding transport="http://schemas.xmlsoap.org/soap/http" style="document" />
    <wsdl:operation name="Email">
      <soap:operation soapAction="http://rn.inf.tu-dresden.de:10080/Email/Email"
       style="document" />
      <wsdl:input>
        <soap:body use="literal" />
      </wsdl:input>
      <wsdl:output>
        <soap:body use="literal" />
      </wsdl:output>
    </wsdl:operation>
  </wsdl:binding>
  <wsdl:service name="EmailService">
    <documentation xmlns="http://schemas.xmlsoap.org/wsdl/">Mail services.</documentation>
    <wsdl:port name="EmailSoap" binding="tns:EmailSoap">
      <soap:address location="http://rn.inf.tu-dresden.de:10080/email.php" />
    </wsdl:port>
  </wsdl:service>
</wsdl:definitions>
```

Listing 5-2: Ausschnitt aus der WSDL des `EmailService` mit `sendEmail`-Methode

Als Eingabeparameter erwartet der Dienst die Angaben zu Absender (`from`), Empfänger (`to`), Betreff (`subject`) und den Text der Nachricht (`message`). Der Rückgabewert ist entweder `success` oder `failure`, je nach Erfolg beim Senden.

Dieser Dienst wurde im Rahmen dieser Arbeit prototypisch implementiert. Als serverseitige Implementierung des E-Mail-Web-Service wird das Programm `sendmail` aufgerufen, um eine E-Mail zu versenden. Auf dem Server läuft ein Framework für Web Services, welches in Python geschrieben ist und ähnlich wie SOAP::Lite[48] ermöglicht, einfache Web Services anhand kleiner Plug-Ins zu betreiben. Dieses Framework wird `TUDWebService` genannt und nimmt SOAP-Anfragen entgegen, die etwa wie folgt aussehen (siehe Listing 5-3).

[48] SOAP::Lite ist ein Perl-Modul für das Erstellen, Versenden und Verarbeiten von SOAP-Meldungen. (http://soaplite.com/)

```
<?xml version='1.0' encoding='utf-8'?>",
  <SOAP-ENV:Envelope
   xmlns:SOAP-ENV='http://schemas.xmlsoap.org/soap/envelope/'
   xmlns:xsi='http://www.w3.org/1999/XMLSchema-instance'
   xmlns:xsd='http://www.w3.org/1999/XMLSchema'>
    <SOAP-ENV:Body>
      <ns1:EmailSoapIn xmlns:ns1='urn:email'
       SOAP-ENV:encodingStyle='http://schemas.xmlsoap.org/soap/encoding/'>
        <to xsi:type='xsd:string'>portalium@gmail.com</to>
        <message xsi:type='xsd:string'>Test</message>
        <from xsi:type='xsd:string'>portalium@gmail.com</from>
        <subject xsi:type='xsd:string'>Eine Testmail</subject>
      </ns1:EmailSoapIn>
    </SOAP-ENV:Body>
  </SOAP-ENV:Envelope>
```

Listing 5-3: SOAP-Anfrage an den E-Mail-Web-Service

Danach übergibt der `TUDWebService` alle gefundenen Variablen an ein passendes Plug-In. Das wird dadurch gelöst, dass jedes Plug-In während der Initialisierung eine Liste von SOAP-*Actions* angibt, die entsprechend auf den HTTP-Header der gesendeten Nachricht zutreffen müssen. Das Plug-In liest die Variablen ein (in dem Fall: `to`, `from`, `message`, `subject`) und überprüft sie auf Vollständigkeit. Dann übergibt es sie direkt an das Programm `sendmail`, welches die Mailzustellung übernimmt. Je nach aufgetretenen Fehlern wird anschließend das Ergebnis zurückgeliefert, also entweder ein Erfolg (`success`) oder ein Misserfolg (`failure`). Die zurück gelieferte SOAP-Nachricht ist in Listing 5-4 dargestellt.

```
<?xml version='1.0' encoding='utf-8'?>
  <soap:Envelope
   xmlns:soap='http://schemas.xmlsoap.org/soap/envelope/'
   xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance'
   xmlns:xsd='http://www.w3.org/2001/XMLSchema'>
    <soap:Body>
      <resultcode>
        success
      </resultcode>
    </soap:Body>
  </soap:Envelope>
```

Listing 5-4: Rückantwort des E-Mail-Web Service

Der E-Mail-Web-Service wurde prototypisch implementiert, womit die Praxistauglichkeit des Ansatzes nachgewiesen werden konnte. Darauf aufbauend kann der Funktionsumfang des E-Mail-Dienstes durch viele weitere nützliche Features ergänzt werden.

5.2.2.2 SMS-Dienst

Ein weiterer möglicher Kommunikationsdienst dient dem Versenden von SMS[49], also von kurzen Textnachrichten. Dieser Dienst wurde in dem öffentlichen Verzeichnis WebServiceX.NET [50] gefunden. Er ermöglicht das kostenlose Versenden von *Short Messages* in viele Länder der Welt. Als Telearbeiter könnte man diesen Dienst nutzen, um einem Kollegen oder Geschäftspartner, der sich gerade auf Reisen befindet, eine wichtige Nachricht zukommen zu lassen. Weiterhin könnte dieser Dienst auch zur automatischen Erzeugung von Bestätigungsnachrichten in komplexen Abläufen wie dem Bearbeiten einer Bestellung (Bestellung eingegangen, Versand ausgelöst, etc.) genutzt werden. Außerdem wären Szenarien zur Kopplung mit einem E-Mail-Dienst (Nachricht, wenn neue E-Mails

[49] SMS (Short Message Service) ist ein Telekommunikationsdienst zur Übertragung von Textnachrichten, der zuerst für den GSM-Mobilfunk entwickelt wurde und nun auch im Festnetz verfügbar ist.

[50] http://www.webservicex.net/

eingetroffen sind) oder dem Kalender-Dienst (Erinnerung an Termine) denkbar. Grundlage des Dienstes ist eine .NET-Applikation, die die SMS versendet. Der Web Service `SendSMSWorld` hat folgende Schnittstellen, deren WSDL-Beschreibung auszugsweise im Listing 5-5 zu sehen ist.

```
<?xml version="1.0" encoding="utf-8" ?>
  <wsdl:definitions xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
   ...
   targetNamespace="http://www.Web ServiceX.NET"
   xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">
    <wsdl:types>
      <s:schema elementFormDefault="qualified"
       targetNamespace="http://www.Web ServiceX.NET">
        <s:element name="sendSMS">
          <s:complexType>
            <s:sequence>
              <s:element minOccurs="0" maxOccurs="1" name="FromEmailAddress"
               type="s:string" />
              <s:element minOccurs="0" maxOccurs="1" name="CountryCode" type="s:string" />
              <s:element minOccurs="0" maxOccurs="1" name="MobileNumber" type="s:string"
               />
              <s:element minOccurs="0" maxOccurs="1" name="Message" type="s:string" />
            </s:sequence>
          </s:complexType>
        </s:element>
        <s:element name="sendSMSResponse">
          <s:complexType>
            <s:sequence>
              <s:element minOccurs="0" maxOccurs="1" name="sendSMSResult" type="s:string"
               />
            </s:sequence>
          </s:complexType>
        </s:element>
        <s:element name="string" nillable="true" type="s:string" />
      </s:schema>
    </wsdl:types>
    <wsdl:message name="sendSMSSoapIn">
      <wsdl:part name="parameters" element="tns:sendSMS" />
    </wsdl:message>
    <wsdl:message name="sendSMSSoapOut">
      <wsdl:part name="parameters" element="tns:sendSMSResponse" />
    </wsdl:message>
    ...
    <wsdl:portType name="SendSMSWorldSoap">
      <wsdl:operation name="sendSMS">
        <documentation xmlns="http://schemas.xmlsoap.org/wsdl/">Send unlimited free
         SMS.</documentation>
        <wsdl:input message="tns:sendSMSSoapIn" />
        <wsdl:output message="tns:sendSMSSoapOut" />
      </wsdl:operation>
    </wsdl:portType>
    ...
    <wsdl:binding name="SendSMSWorldSoap" type="tns:SendSMSWorldSoap">
      <soap:binding transport="http://schemas.xmlsoap.org/soap/http" style="document" />
      <wsdl:operation name="sendSMS">
        <soap:operation soapAction="http://www.Web ServiceX.NET/sendSMS" style="document"
         />
        <wsdl:input>
          <soap:body use="literal" />
        </wsdl:input>
        <wsdl:output>
          <soap:body use="literal" />
        </wsdl:output>
      </wsdl:operation>
    </wsdl:binding>
    ...
    <wsdl:service name="SendSMSWorld">
      <documentation xmlns="http://schemas.xmlsoap.org/wsdl/" />
      <wsdl:port name="SendSMSWorldSoap" binding="tns:SendSMSWorldSoap">
        <soap:address location="http://www.Web Servicex.com/sendsmsworld.asmx" />
      </wsdl:port>
      <wsdl:port name="SendSMSWorldHttpGet" binding="tns:SendSMSWorldHttpGet">
        <http:address location="http://www.Web Servicex.com/sendsmsworld.asmx" />
      </wsdl:port>
     <wsdl:port name="SendSMSWorldHttpPost" binding="tns:SendSMSWorldHttpPost">
```

```
        <http:address location="http://www.Web Servicex.com/sendsmsworld.asmx" />
      </wsdl:port>
    </wsdl:service>
  </wsdl:definitions>
</wsdl:definitions>
```

Listing 5-5: Auszug aus der WSDL des Web Service `SendSMSWorld`

Ein ganz ähnlicher Dienst zum Versenden von Fax-Nachrichten konnte ebenfalls auf WebServiceX.NET gefunden werden. Dieser kann für Telearbeiter sehr praktisch sein, da sie nicht immer über ein Faxgerät verfügen, besonders wenn sie mobile Telearbeit ausüben. Auch für alle anderen Kommunikationsdienste sind solche Implementierungen als Web Services ebenfalls denkbar und teilweise auch schon vorhanden, aber bisher nicht öffentlich verfügbar.

5.2.3 Kollaborationsdienste

Als Kollaborationsdienste werden alle Dienste verstanden, die die Zusammenarbeit aller an einer Aufgabe beteiligten Personen gewährleisten. Räumlich verteilte Kollaboration setzt voraus, dass die Daten der gemeinsamen Informationsbasis von verschiedenen Standorten aus bearbeitbar sind, sowohl von einer Person als auch gemeinsam von mehreren Personen.

Wichtigste Vertreter dieser Gruppe sind Anwendungen zum Dokumentenaustausch und zur Dokumentenverwaltung. Dokumente sind tragende Säulen für den inner- und zwischenbetrieblichen Informationsfluss und für die Informationsspeicherung. So vielfältig sich die verschiedenen Informationen darstellen, so verschiedenartig sind die Erscheinungsformen der Dokumente. Sie reichen vom einfachen Memo mit kurzer Verfallszeit, über Geschäftsbriefe und Verträge mit rechtlicher Bedeutung und mittleren Aufbewahrungsfristen, bis hin zu umfangreichen Dokumentationen, die z. B. den Entstehungsprozess eines Produkts über seine ganze Lebensdauer hinweg dokumentieren. Eine auf die betrieblichen Anforderungen zugeschnittene Dokumentenverwaltung ermöglicht es, die benötigten Dokumente rasch und ökonomisch zu erfassen, zu indizieren und abzulegen, um sie später zuverlässig und schnell wieder aufzufinden. Sie hilft auch, die Kosten für die Verteilung, Verwaltung und Archivierung der Dokumente in vernünftigen Grenzen zu halten.

Weitere wichtige Kollaborationsdienste sind Dienste zum Informations- und Wissensmanagement. *Wissensmanagementsysteme* (WMS) bilden und verwalten die Wissensbasis eines Unternehmens. Unter der Wissensbasis eines Unternehmens werden alle Daten und Informationen, alles Wissen und alle Fähigkeiten verstanden, die diese Organisation zur Lösung ihrer vielfältigen Aufgaben benötigt. Dabei werden individuelles Wissen und persönliche Fähigkeiten (*Human Capital*) durch die Aufbereitung in Informationssystemen in strukturelles, also in der Organisation verankertes, Wissen transformiert.

Hauptziel dieser Dienste ist folglich die Verwaltung und Speicherung des Wissens und der Kompetenzen der Mitarbeiter, um effektiveres Arbeiten zu ermöglichen. Dazu gibt es auf dem Markt sehr komplexe *Content Management Systeme* (CMS), die über Redaktionssysteme mit Wissen gefüllt werden können, aber auch einfache so genannte Wiki-Systeme[51], die eine direkte Bearbeitung von Web Seiten im Browser ermöglichen und damit die gemeinsame Bearbeitung von Informationen und Wissensbasen erlauben. Dabei können alle zugelassenen Leser auch gleichzeitig Redakteure der Inhalte sein. Dies erleichtert die Sammlung von Wissen und Kompetenzen im Unternehmen.

[51] Wiki, abgeleitet vom Hawaiianischen „wiki wiki“ (schnell), ist eine Technologie zum Management von Wissensbasen, die aus mehreren verlinkten Hypertexten bestehen, die direkt im Browser bearbeitet werden können.

Solche Dienste stellen für den Telearbeiter, der meist von der direkten Kommunikation im Unternehmen ausgeschlossen ist und nur begrenzte Möglichkeiten zur Nachfrage bei Kollegen hat, eine wichtige Wissensquelle bei der Arbeit dar. Deshalb sollte sich die Unternehmenskultur dahingehend entwickeln, dass Informationen und Wissen, das für viele Mitarbeiter relevant sein könnte, mittels solcher Systeme gesammelt und verwaltet werden. In den vergangenen Jahren hat auch in den Unternehmensleitungen ein Umdenken stattgefunden und der Grundsatz, dass das Wissen der Mitarbeiter ein wichtiger Produktivitätsfaktor ist, gehört mittlerweile zu einer modernen und erfolgreichen Managementphilosophie.

5.2.3.1 Dokumentenmanagementdienst

Ein sehr wichtiger Dienst zur Unterstützung von Telearbeit ist ein Dokumentenmanagementdienst, der dem Telearbeiter Zugriff auf alle benötigten Dokumente zur Erledigung seiner Aufgaben ermöglicht. Vor allem, wenn der Telearbeiter in ein Projektteam integriert ist, ist eine gruppenweite Dokumentenverwaltung sehr wichtig.

Als zentraler Dokumentenmanagementdienst wird ein Web Service zur Verfügung gestellt, der vielfältige Methoden zum Zugriff auf die Dokumentenbasis einer Arbeitsgruppe oder des ganzen Unternehmens anbietet. Dabei können z.B. einfache Methoden zum Abruf des Status eines Dokumentes angeboten werden, die auch von einem mobilen Endgerät aufgerufen werden können. So wäre es denkbar, dass sich ein Telearbeiter von unterwegs aus mit Hilfe seines Mobiltelefons kurz über den Bearbeitungsstand eines Dokumentes informieren möchte. Dann kann er eine `GetStatusDocument`-Anfrage an den Dokumentenmanagementdienst unter Angabe des Dokumentennamens stellen und als Antwort den Bearbeitungsstatus und -zeitpunkt sowie den Namen des letzten Bearbeiters erfahren. Wenn er dann an seinem Arbeitsplatz (ob zu Hause oder im Unternehmen) angekommen ist, kann er über andere Methoden auf das Dokument zugreifen und es bearbeiten oder sich mit dem letzten Bearbeiter in Verbindung setzen, um weitere Änderungen zu besprechen. Weitere vielfältige Anwendungsszenarien wären hier denkbar.

Folgende grundlegende Funktionalitäten sollten durch einen Dokumentenmanagement-Service angeboten werden:

- **Dokument ablegen**: Mit dieser Operation wird das manuelle Einladen der Dokumente in das DMS modelliert. Bei der Eingabe eines neuen Dokumentes sind vergleichsweise viele Benutzeraktionen und Systemprozesse notwendig. Zunächst muss die vom Nutzer ausgewählte Datei ins Archiv kopiert werden. Dann wird für das Dokument eine neue Ressource erzeugt und die entsprechenden Metadaten vorbelegt. Danach kann das Dokument indiziert werden, dies kann sowohl manuell als auch automatisch erfolgen.
- **Dokument indizieren**: Mit dieser Operation können bereits vorbelegte Metadaten bearbeitet und neue Metadaten hinzugefügt werden. Dabei sollten die Daten zuerst auf Zulässigkeit geprüft werden, bevor sie ins System eingetragen werden.
- **Dokument suchen**: Ein DMS sollte sowohl eine strukturierte als auch eine Volltextsuche anbieten. Als Grundlage für die strukturierte Suche dienen die Metadaten, die bei der Indizierung erzeugt wurden.
- **Dokument auschecken**: Diese Operation wird benötigt, wenn ein Telearbeiter ein Dokument bearbeiten will und verhindert werden soll, dass andere Mitarbeiter zur gleichen Zeit Änderungen vornehmen. Nachdem das Dokument ausgecheckt wurde, ist es für weitere Änderungen gesperrt.
- **Dokument einchecken**: Nach der Bearbeitung muss das Dokument wieder eingecheckt und damit für die Bearbeitung durch andere freigegeben werden. Unterstützt das DMS auch die Versionierung von Dokumenten, muss die aktuelle Version des Doku-

ments als "veraltet" markiert und die neue Version des Dokuments abgelegt und indiziert werden.

- **Dokument ansehen**: Diese Operation ermöglicht den lesenden Zugriff auf ein Dokument, ohne dass es für die Bearbeitung durch andere gesperrt werden muss. Dabei wird die Datei aus dem Archiv ins Temp-Verzeichnis kopiert und in einem externen Viewer angezeigt.
- **Status eines Dokumentes abfragen**: Vor allem in mobilen Szenarien kann aufgrund beschränkter Bandbreite oder zu kleiner Displays oft nicht ein ganzes Dokument ausgelesen werden. Dann ist es oft ausreichend, wenn bestimmte Statusinformationen wie Bearbeitungszeit, -status und letzter Bearbeiter ausgegeben werden. Welche konkreten Informationen gewünscht werden, können als Parameter an den Web Service übergeben werden.
- **Dokument bearbeiten**: Diese Operation stellt eine komplexe Operation dar, da sie mehrere Basisoperationen integriert. Zum Bearbeiten eines Dokumentes muss dieses zuerst ausgecheckt werden, um dann eine externe Anwendung zum Editieren zu starten. Nach Abschluss der Bearbeitung wird das Dokument wieder eingecheckt.
- **Dokument löschen**: Mit Hilfe dieser Operation können nicht mehr benötigte Dokumente und dazugehörige Metadaten aus dem System entfernt werden.

Die genannten Operationen sind noch sehr allgemein beschrieben, erfassen aber zunächst die grundlegende Funktionalität, die der Dokumentenmanagementdienst anbieten soll. Die Verfeinerung der Methoden wird im Folgenden mit der Implementierung dieses Dienstes detaillierter beschrieben.

Am Beispiel „Dokument bearbeiten“ ist zu erkennen, dass auch komplexere Abläufe abgebildet werden müssen, was eine Komposition mehrerer Operationen notwendig macht. Deshalb wurden in der Implementierung zwei verschiedene Klassen für die Definition der Schnittstellen realisiert. `SimpleDocumentManager` enthält alle Basismethoden zur Dokumentenverwaltung, wohingegen `ComplexDocumentManager` komplexere Methoden implementiert (siehe Abbildung 5-5). So ist beispielsweise die *Checkout*-Methode auch durch mehrere Basisoperationen modellierbar, jedoch müsste der Dienstnutzer diese dann selbst kombinieren. Die komplexe Operation bietet dem Nutzer dagegen die gesamte Funktionalität über eine vereinfachte Schnittstelle an.

Die Methoden, die durch die Schnittstellen implementiert werden, stehen zum Teil in verschiedenen Ausprägungen zur Verfügung. So muss bei der Ausgabe eines Dokuments unterschieden werden, ob das gesamte Dokument als File oder nur eine Referenz darauf übertragen werden soll. Dies kann zum Beispiel abhängig vom Endgerät des Anwenders sein. Wird lediglich ein URI zurückgegeben, muss dem Benutzer das Dokument unter der spezifizierten URI für einen definierten Zeitraum zur Verfügung stehen. Gegebenenfalls reicht es auch aus, wenn nur Statusinformationen über das Dokument, z.B. das Datum der letzten Änderung und der Name des Bearbeiters, ermittelt werden. Die gesendeten Informationen sollten dabei nicht die komplett verfügbaren Attribute umfassen, sondern abhängig vom Ausgabegerät bzw. von den persönlichen Anforderungen des Benutzers nur einen Teil der Dokument-Informationen abbilden. Damit nicht bei jedem Request die Art des Ausgabegerätes bzw. die individuelle Konfiguration übermittelt werden müssen, wäre es denkbar, diese Preferenzen zentral, möglichst beim Login in das System, abzuspeichern und damit den Zugriff des Nutzers zu personalisieren.

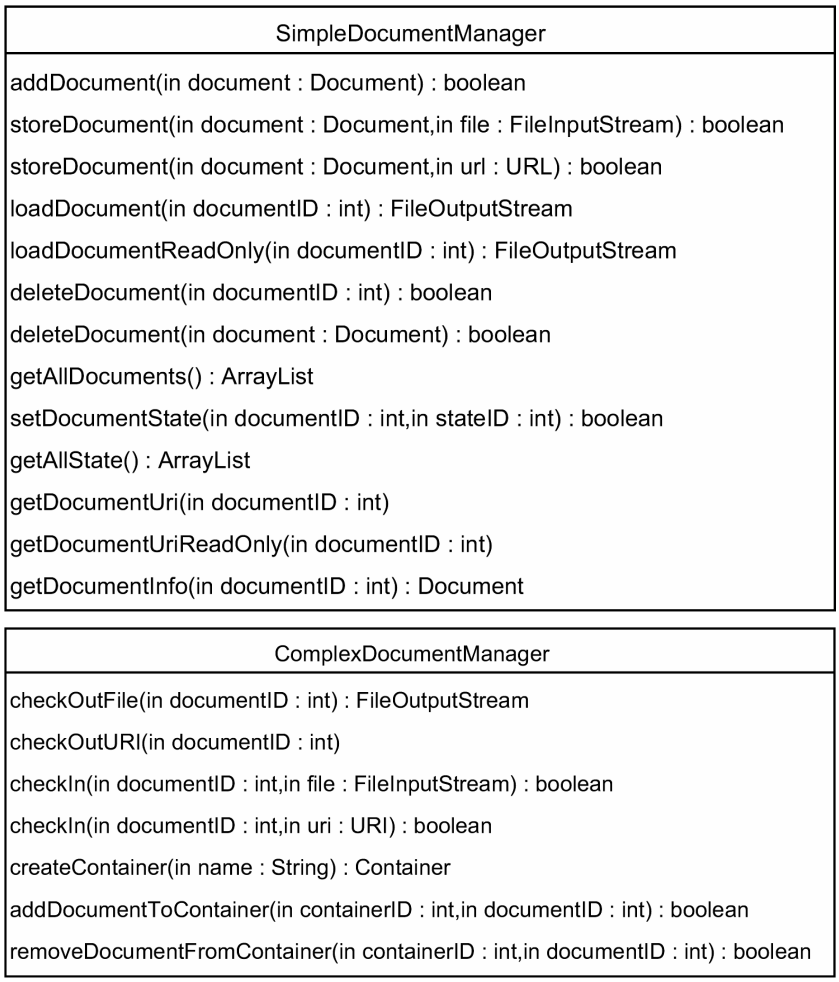

Abbildung 5-5: Klassendiagramm der DocumentManager-Schnittstellen

Ausgehend von der in Abbildung 5-5 dargestellten Klassendefinition ergeben sich zwei Schnittstellenbeschreibungen im WSDL-Format für jedes Paket. An dieser Stelle soll lediglich am Beispiel einer ausgewählten Methode (`checkOutURI`) ein Auszug aus den generierten WSDL-Dateien gezeigt werden. Die WSDL-Beschreibungen aller Methoden der beiden Klassen `SimpleDocumentManager` und `ComplexDocumentManager` befinden sich im Anhang (Listing A-1 und A-2).

```
<wsdl:definitions ...>
  <wsdl:types>...</wsdl:types>
  <wsdl:message name="checkOutURIRequest">
    <wsdl:part name="documentID" type ="xsd:int"/>
  </wsdl:message>
  <wsdl:message name="checkOutURIResponse">
    <wsdl:part name="checkOutURIReturn" type="tns1:URI"/>
  </wsdl:message>
  <wsdl:portType name="ComplexDocumentManager">
    <wsdl:operation name="checkOutURI" parameterOrder="documentID">
      <wsdl:input name="checkOutURIRequest" message="impl:checkOutURIRequest"/>
      <wsdl:output name="checkOutURIResponse" message="impl:checkOutURIResponse"/>
    </wsdl:operation>
  </wsdl:portType>
  <wsdl:binding name ="ComplexDocumentManagerSoapBinding"
   type="impl:ComplexDocumentManager">
    <wsdlsoap:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http"/>
    <wsdl:operation name ="checkOutURI">
      <wsdlsoap:operation soapAction=""/>
      <wsdl:input name ="checkOutURIRequest">
        <wsdlsoap:body use="encoded" encodingStyle="http://schemas.xmlsoap.org/soap/
         encoding/" namespace="http://documentManager.doc"/>
      </wsdl:input>
      <wsdl:output name="checkOutURIResponse">
```

```
        <wsdlsoap:body use="encoded" encodingStyle="http://schemas.xmlsoap.org/soap/
         encoding/" namespace="http://documentManager.doc"/>
      </wsdl:output>
    </wsdl:operation>
  </wsdl:binding>
  <wsdl:service name="ComplexDocumentManagerService">
    <wsdl:port name="ComplexDocumentManager"
     binding="impl:ComplexDocumentManagerSoapBinding">
      <wsdlsoap:address location=http://localhost:8080/dms/services/ComplexDocumentManager
       />
    </wsdl:port>
  </wsdl:service>
</wsdl:definitions>
```

Listing 5-6: WSDL der Operation `CheckOutURI` des Dokumentenmanagementdienstes

Neben diesen Funktionen zur Verwaltung und zum Zugriff auf Dokumente werden auch Methoden zur Administration, z.B. zur Nutzerverwaltung und zum Monitoring, benötigt. Diese könnten beispielsweise über eine zentrale Admin-Anwendung analog einem Kontrollzentrum aufgerufen werden. Die gesamte Datenhaltung wurde auf Basis von RDF bzw. OWL modelliert. Dazu wurden die in [Abe03] vorgestellten Ontologien verwendet und gegebenenfalls an die eigenen Anforderungen angepasst. Zur Beschreibung der Dokumente und Verwaltung der Metadaten wurde eine Dokument-Ontologie erstellt. In der System-Ontologie wurden dagegen Klassen zur Nutzerverwaltung und Administration beschrieben. Die Dokument-Ontologie wird zur Beschreibung von Dokumenten und deren Eigenschaften verwendet. Beispielsweise gehören die Klassen `Document`, `State` und `Format` in diese Definition. Listing 5-7 zeigt einen Ausschnitt aus der OWL/XML-Notation dieser Ontologie.

```
<!DOCTYPE rdf:RDF [
  <!ENTITY xsd "http://www.w3.org/2001/XMLSchema#" >
  <!ENTITY dc "http://purl.org/dc/elements/1.1/#" >
  <!ENTITY doc "http://www.inf.tu-dresden.de/~jh182921/document#" > ]>
<rdf:RDF ...>
  <owl:Ontologie about="">
    <dc:title>Datenformate fuer Dokumenten-Management-Systeme</dc:title>
    <dc:subject>DMS</dc:subject>
    <dc:description>Eine OWL-Version fuer die Beschreibung des Datenbestandes eines
     DMS</dc:description>
    <dc:identifier>http://rn.inf.tu-dresden.de/dms.owl</dc:identifier>
  </owl:Ontologie>
  <!-- Property Document-->
  <owl:DatatypeProperty rdf:ID="title">
    <rdfs:domain rdf:resource="#Document"/>
    <rdfs:range rdf:resource="&dc;title"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:ID="description">
    <rdfs:domain rdf:resource="#Document"/>
    <rdfs:range rdf:resource="&dc;description"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:ID="creationDate">
    <rdfs:domain rdf:resource="#Document"/>
    <rdfs:range rdf:resource="&dc;date"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:ID="validDate">
    <rdfs:domain rdf:resource="#Document"/>
    <rdfs:range rdf:resource="&dc;date"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:ID="identifier">
    <rdfs:domain rdf:resource="#Document"/>
    <rdfs:range rdf:resource="&dc;identifier"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:ID="creator">
    <rdfs:domain rdf:resource="#Document"/>
    <rdfs:range rdf:resource="&dc;creator"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:ID="fileName">
    <rdfs:domain rdf:resource="#Document"/>
    <rdfs:range rdf:resource="&xsd;string"/>
  </owl:DatatypeProperty>
  <!-- ObjectProperty Document -->
```

```
  <owl:ObjectProperty rdf:ID="subject">
    <rdfs:domain rdf:resource="#Document"/>
    <rdfs:range rdf:resource="#Subject"/>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:ID="language">
    <rdfs:domain rdf:resource="#Document"/>
    <rdfs:range rdf:resource="#Language"/>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:ID="state">
    <rdfs:domain rdf:resource="#Document"/>
    <rdfs:range rdf:resource="#State"/>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:ID="format">
    <rdfs:domain rdf:resource="#Document"/>
    <rdfs:range rdf:resource="#Format"/>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:ID="type">
    <rdfs:domain rdf:resource="#Document"/>
    <rdfs:range rdf:resource="#Type"/>
  </owl:ObjectProperty>
  <!-- State Property-->
  <owl:DatatypeProperty rdf:ID="stateID">
    <rdfs:domain rdf:resource="#State"/>
    <rdfs:range rdf:resource="&xsd;nonNegativeInteger"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:ID="stateName">
    <rdfs:domain rdf:resource="#State"/>
    <rdfs:range rdf:resource="&xsd;string"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:ID="stateDescription">
    <rdfs:domain rdf:resource="#State"/>
    <rdfs:range rdf:resource="&xsd;string"/>
  </owl:DatatypeProperty>
</rdf:RDF>
```

Listing 5-7: Ausschnitt aus der OWL-Beschreibung der Dokument-Ontologie

Abbildung 5-6 stellt die Zusammenhänge der Dokument-Ontologie graphisch dar.

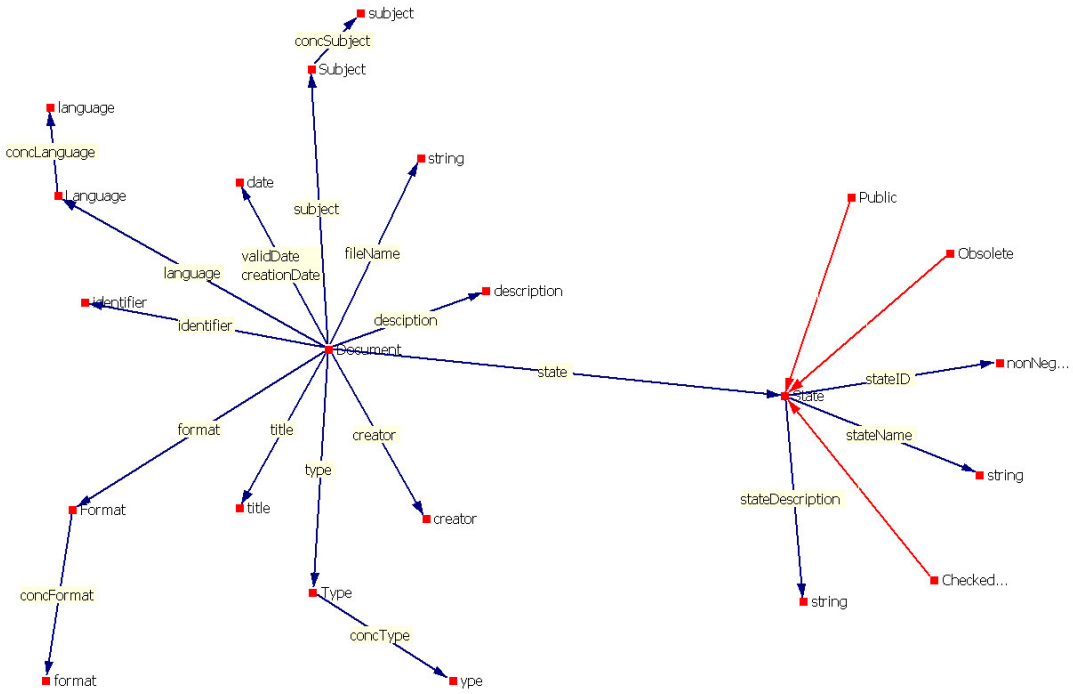

Abbildung 5-6: Graph der Dokument-Ontologie

Für die permanente Speicherung der Dokumente und Metadaten in einer Datenbank sollte aufgrund der XML-lastigen Notation der Einsatz einer XML-Datenbank bevorzugt werden.

Als Anfragesprache könnte dann RDQL[52] genutzt werden, da das Datenbankschema dann durch die OWL-Beschreibung vorgegeben wird.

5.2.4 Koordinationsdienste

Unter Koordinationsdiensten werden all jene Dienste verstanden, die die Zusammenarbeit in einer Gruppe koordinieren. Dazu gehören Dienste zur Organisation von Projekten, zur Bildung von Teams und zur Verteilung und Abrechnung von Arbeitsaufgaben. Auch die gemeinsame Terminplanung und -verwaltung ist ein wichtiger Koordinationsdienst. Solche Funktionen werden bisher vor allem von Projektmanagementsystemen, Workflow-managementsystemen oder auch Groupware-Lösungen angeboten. Diese sind meist aber sehr komplexe Systeme mit einer Fülle ungenutzter Funktionalität und einem großen Verwaltungsoverhead. Deshalb ist es sinnvoll, einzelne Funktionalitäten als Dienste anzubieten, die entsprechend flexibel kombiniert werden können.

5.2.4.1 Terminverwaltungs-/Kalenderdienst

Als Beispiel für einen Koordinationsdienst soll hier kurz ein Web Service namens `CalendarService` beschrieben werden, der die Verwaltung von Terminen ermöglicht.

Zu den Grundfunktionalitäten, die ein Kalenderdienst anbieten soll, gehören folgende:

- Termine eintragen, ändern, löschen,
- Abfragen: Termin frei oder belegt, alle Termine eines bestimmten Zeitraumes, nächster Termin,
- Automatische Erinnerung an Termine.

Weitere Zusatzfunktionen wären wünschenswert:

- wiederkehrende Termine/Terminserien verwalten,
- verschiedene Gruppen von Terminen unterscheiden: privat, dienstlich, ...,
- unterschiedliche Prioritäten von Terminen festlegen,
- Synchronisation mit Kalendern anderer Personen oder mit weiteren eigenen Terminverwaltungssystemen,
- Gruppenkalenderfunktionen wie Besprechungsplanung etc. .

Um diese verschiedenen Funktionalitäten anbieten zu können, muss der `CalendarService` mehrere Operationen mit unterschiedlichen Ein- und Ausgabeparametern zur Verfügung stellen. Die Beschreibung der Schnittstellen mittels WSDL für die Operation `AddEvent` zum Hinzufügen eines Termins ist im folgenden Listing 5-8 zu sehen.

```
<?xml version="1.0" encoding="utf-8" ?>
<wsdl:definitions xmlns:s="http://www.w3.org/2001/XMLSchema"
 xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
 xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
 xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
 xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/"
 xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
 xmlns:tns="http://rn.inf.tu-dresden.de/xmlservices/Calendar"
 targetNamespace="http://rn.inf.tu-dresden.de/xmlservices/Calendar">
  <wsdl:types>
    <s:schema elementFormDefault="qualified"
     targetNamespace="http://rn.inf.tu-dresden.de/xmlservices/Calendar">
      <s:element name="CalendarResponse">
        <s:complexType>
          <s:sequence>
```

[52] RDQL (RDF Data Query Language) ist eine vom W3C standardisierte SQL-ähnliche Anfragesprache für RDF-Graphen. (http://www.w3.org/Submission/RDQL/)

```
            <s:element minOccurs="0" maxOccurs="1" name="resultcode" type="s:string" />
          </s:sequence>
        </s:complexType>
      </s:element>
      <s:element name="CalendarRequest">
        <s:complexType>
          <s:sequence>
            <s:element minOccurs="0" maxOccurs="1" name="Event" type="tns:Event" />
            <s:element minOccurs="1" maxOccurs="1" name="userID" type="tns:string" />
            <s:element minOccurs="1" maxOccurs="1" name="password" type="tns:string" />
          </s:sequence>
        </s:complexType>
      </s:element>
      <s:complexType name="Event">
        <s:sequence>
          <s:element minOccurs="1" maxOccurs="1" name="date" type="tns:date" />
          <s:element minOccurs="1" maxOccurs="1" name="startTime" type="tns:time" />
          <s:element minOccurs="1" maxOccurs="1" name="endTime" type="tns:time" />
          <s:element minOccurs="1" maxOccurs="1" name="place" type="tns:string" />
          <s:element minOccurs="1" maxOccurs="1" name="subject" type="s:string" />
          <s:element minOccurs="0" maxOccurs="1" name="notes" type="s:string" />
          <s:element minOccurs="0" maxOccurs="1" name="remindPeriod" type="s:string" />
        </s:sequence>
      </s:complexType>
    </s:schema>
  </wsdl:types>
  <wsdl:message name="CalendarSoapIn">
    <wsdl:part name="parameters" element="tns:CalendarRequest" />
  </wsdl:message>
  <wsdl:message name="CalendarSoapOut">
    <wsdl:part name="parameters" element="tns:CalendarResponse" />
  </wsdl:message>
  <wsdl:portType name="CalendarSoap">
    <wsdl:operation name="AddEvent">
      <documentation xmlns="http://schemas.xmlsoap.org/wsdl/">Add an event to your
       calendar.</documentation>
      <wsdl:input message="tns:CalendarSoapIn" />
      <wsdl:output message="tns:CalendarSoapOut" />
    </wsdl:operation>
  </wsdl:portType>
  <wsdl:binding name="CalendarSoap" type="tns:CalendarSoap">
    <soap:binding transport="http://schemas.xmlsoap.org/soap/http" style="document" />
    <wsdl:operation name="AddEvent">
      <soap:operation soapAction="http://rn.inf.tu-dresden.de:10080/Calendar/AddEvent"
       style="document" />
      <wsdl:input>
        <soap:body use="literal" />
      </wsdl:input>
      <wsdl:output>
        <soap:body use="literal" />
      </wsdl:output>
    </wsdl:operation>
  </wsdl:binding>
  <wsdl:service name="CalendarService">
    <documentation xmlns="http://schemas.xmlsoap.org/wsdl/">Calendar services.
    </documentation>
    <wsdl:port name="CalendarSoap" binding="tns:CalendarSoap">
      <soap:address location="http://rn.inf.tu-dresden.de:10080/Calendar.php" />
    </wsdl:port>
  </wsdl:service>
</wsdl:definitions>
```

Listing 5-8:WSDL der Operation `AddEvent` des `CalendarService`

Neben den eigentlichen Termindaten im komplexen Typ `Event` werden auch eine Nutzer-ID und ein Passwort als Eingabeparameter übergeben, um sicherzustellen, dass nur berechtigte Nutzer Termine eintragen können, und um zu identifizieren, welchem Kalender der Termin zuzuordnen ist. Es wäre denkbar, dass der Provider des Kalenderdienstes ein externes Unternehmen ist, das diesen Service entgeltlich verschiedenen Nutzern zur Verfügung stellt. Deshalb muss er dafür Sorge tragen, dass die verschiedenen Kalender entsprechend vor unberechtigtem Zugriff geschützt werden. Um die oben genannten Zusatzfunktionen zu ermöglichen, wären weitere Eingabeparameter wie die Art des Termins (privat oder geschäftlich) oder die Priorität, die der Termin hat, notwendig. Analog

können die Schnittstellen der anderen Operationen des `CalendarService` beschrieben werden.

Ein entscheidender Vorteil der Implementierung als Web Service ist die Möglichkeit, auf Kalenderdaten mit ganz verschiedenen Clients und Endgeräten zuzugreifen. So wäre es denkbar, dass sich ein mobiler Telearbeiter unterwegs seine nächsten Termine in Kurzform auf das Handy lädt, wogegen er zu Hause oder im Büro angekommen, die volle Funktionalität des Kalenders über eine Weboberfläche nutzen kann.

5.2.5 Sicherheitsdienste

Die Sicherheitsdienste sind für die Bereitstellung von Sicherheitsfunktionalität auf allen Ebenen der Architektur verantwortlich. Dazu gehören die Benutzerverwaltung und Authentifizierung auf Anwendungsebene ebenso wie die Sicherung der Datenübertragung auf Netzwerkebene. Auch die Verschlüsselung firmeninterner Daten und der Schutz personenbezogener Daten sind wichtige Aufgaben von Sicherheitsdiensten. Dabei können diese Dienste auf vorhandene Protokolle und Anwendungen zurückgreifen, die z.B. durch das Betriebssystem oder Netzwerkprotokolle realisiert werden. Eine Reihe von Sicherheitsfunktionen wird auch durch die Portalengine übernommen. Diese bietet z.B. Funktionen zur Authentifizierung und zum Single-Sign-On.

Weitere Dienste für digitales Signieren oder Verschlüsseln können auch durch externe Service Provider erbracht werden, wenn ein entsprechendes Vertrauensverhältnis gegeben ist. Weiterhin zählt das Monitoring und Logging der Zugriffe in allen Schichten (z.B. der Dienst- und Dokumentenzugriffe) zur Klasse der Sicherheitsdienste. Bei der Integration dieser Sicherheitsbasisdienste in komplexe Dienste ist zu beachten, dass sich verschiedene Dienste behindern oder sogar negieren können, wenn die damit verfolgten Schutzziele konträr sind (z.B. Privatheit contra Verbindlichkeit). Solche Zusammenhänge könnten mit Hilfe einer sicherheitsspezifischen Ontologie beschrieben und bei der Komposition berücksichtigt werden. Im Folgenden werden exemplarisch einige Sicherheitsdienste vorgestellt und deren Schnittstellen und Implementierung beschrieben.

5.2.5.1 Verschlüsselungsdienst

Dieser Dienst hat die Aufgabe, Dokumente und Nachrichten zu verschlüsseln, um die Vertraulichkeit der Daten zu garantieren. Dabei können sowohl symmetrische als auch asymmetrische Verschlüsselungsverfahren zum Einsatz kommen. In der Regel werden aber meist beide Mechanismen parallel benutzt. Ein gängiges Verfahren besteht z.B. darin, das wesentlich langsamere asymmetrische Verfahren nur für die Übertragung eines symmetrischen Schlüssels zu nutzen, und das eigentliche Dokument dann symmetrisch zu verschlüsseln. So könnte man das Dokument mit Hilfe eines Triple-DES-Schlüssels kodieren und diesen Schlüssel dann wiederum mit RSA sichern. Die Auswahl des Mechanismus kann der Dienstnutzer durch die Vorgabe eines bestimmten Mechanismus beeinflussen. Mit Hilfe von *XML Encryption* [W3C02c] wäre es darüber hinaus auch möglich, nur Teile eines Dokumentes zu verschlüsseln, sofern es sich dabei um ein XML-Dokument handelt. Als Eingabeparameter für den Dienst sind Folgende notwendig: Dokument, Mechanismus und öffentlicher Schlüssel des Empfängers. Als Ausgabeparameter gibt der Dienst das verschlüsselte Dokument zurück. Wie die genaue Schnittstellenbeschreibung eines solchen Verschlüsselungsdienstes aussehen kann, ist in Listing 5-9 zu sehen.

```
<?xml version="1.0" encoding="UTF-8" ?>
<wsdl:definitions xmlns:s="http://www.w3.org/2001/XMLSchema"
 xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
 xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
 xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
```

```
 xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/"
 xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" xmlns:tns="http://rn.inf.tu-dresden.de/
  xmlservices/EncryptionService"
 targetNamespace="http://rn.inf.tu-dresden.de/xmlservices/EncryptionService">
  <wsdl:types>
    <s:schema elementFormDefault="qualified" targetNamespace="http://rn.inf.tu-dresden.de/
     xmlservices/EncryptionService">
      <s:element name="EncryptionServiceResponse">
        <s:sequence>
          <s:element minOccurs="1" maxOccurs="1" name="encryptedDocument" type="tns:file"
           />
        </s:sequence>
      <s:element name="EncryptionServiceRequest">
        <s:complexType>
          <s:sequence>
            <s:element minOccurs="0" maxOccurs="1" name="message"
             type="tns:EncryptionInput" />
          </s:sequence>
        </s:complexType>
      </s:element>
      <s:complexType name="EncryptionInput">
        <s:sequence>
          <s:element minOccurs="1" maxOccurs="1" name="document" type="tns:file" />
          <s:element minOccurs="1" maxOccurs="1" name="mechanism" type="tns:string" />
          <s:element minOccurs="1" maxOccurs="1" name="key" type="tns:binary" />
        </s:sequence>
      </s:complexType>
    </s:schema>
  </wsdl:types>
  <wsdl:message name="EncryptionServiceSoapIn">
    <wsdl:part name="parameters" element="tns:EncryptionServiceRequest" />
  </wsdl:message>
  <wsdl:message name="EncryptionServiceSoapOut">
    <wsdl:part name="parameters" element="tns:EncryptionServiceResponse" />
  </wsdl:message>
  <wsdl:portType name="EncryptionServiceSoap">
    <wsdl:operation name="EncryptionService">
      <documentation xmlns="http://schemas.xmlsoap.org/wsdl/">Encoding a document.
      </documentation>
      <wsdl:input message="tns:EncryptionServiceSoapIn" />
      <wsdl:output message="tns:EncryptionServiceSoapOut" />
    </wsdl:operation>
  </wsdl:portType>
  <wsdl:binding name="EncryptionServiceSoap" type="tns:EncryptionServiceSoap">
    <soap:binding transport="http://schemas.xmlsoap.org/soap/http" style="document" />
    <wsdl:operation name="EncryptionService">
      <soap:operation soapAction="http://rn.inf.tu-dresden.de:10080/EncryptionService/
       EncryptionService" style="document" />
      <wsdl:input>
        <soap:body use="literal" />
      </wsdl:input>
      <wsdl:output>
        <soap:body use="literal" />
      </wsdl:output>
    </wsdl:operation>
  </wsdl:binding>
  <wsdl:service name="EncryptionService">
    <documentation xmlns="http://schemas.xmlsoap.org/wsdl/">Mail services.</documentation>
    <wsdl:port name="EncryptionServiceSoap" binding="tns:EncryptionServiceSoap">
      <soap:address location="http://rn.inf.tu-dresden.de:10080/EncryptionService.php" />
    </wsdl:port>
  </wsdl:service>
</wsdl:definitions>
```

Listing 5-9: WSDL-Beschreibung des `EncryptionService`

Da der Dienstanbieter aber das Original-Dokument erhält, muss er sich in einer vertrauenswürdigen Domäne befinden, z.B. im eigenen Unternehmen. Ansonsten würde die Übertragung des Dokuments zum Dienstanbieter wieder eine Sicherheitslücke darstellen. Hier könnte die Nutzung von *XML Encryption* zur Verschlüsselung der SOAP-Nachrichten Abhilfe schaffen. Dieses Protokoll und weitere Sicherheitsmechanismen für Web Services wurden im Standard *WS-Security* zusammengefasst und von OASIS standardisiert [OAS04a].

5.2.5.2 Signierungsdienst

Dieser Sicherheitsdienst hat die Aufgabe, das Signieren von Dokumenten durchzuführen. Mit einer Signatur lassen sich Authentizität, Integrität und Nicht-Anfechtbarkeit umsetzen. Um die Signatur auch überprüfen zu können, benötigt man den öffentlichen Schlüssel des Dienstanbieters und ein Zertifikat einer vertrauenswürdigen *Certificate Authority* (CA), das belegt, dass dieser Schlüssel auch zu diesem Anbieter gehört.

Als Eingabe erwartet der Signierungsdienst folglich das zu signierende Dokument. Als Ausgabe werden das signierte Dokument, der öffentliche Schlüssel zum Lesen der Signatur und das Zertifikat des Dienstanbieters zurückgeliefert. Die Schnittstellenbeschreibung des `SigningService` ist in Listing 5-10 dargestellt.

```
<?xml version="1.0" encoding="UTF-8" ?>
<wsdl:definitions xmlns:s="http://www.w3.org/2001/XMLSchema"
 xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
 xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
 xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
 xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/"
 xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" xmlns:tns="http://rn.inf.tu-dresden.de/
  xmlservices/SigningService"
 targetNamespace="http://rn.inf.tu-dresden.de/xmlservices/SigningService">
  <wsdl:types>
    <s:schema elementFormDefault="qualified" targetNamespace="http://rn.inf.tu-dresden.de/
     xmlservices/SigningService">
      <s:element name="SigningServiceRequest">
        <s:sequence>
          <s:element minOccurs="1" maxOccurs="1" name="document" type="tns:file" />
        </s:sequence>
      <s:element name="SigningServiceResponse">
        <s:complexType>
          <s:sequence>
            <s:element minOccurs="0" maxOccurs="1" name="message" type="tns:SigningOutput"
             />
          </s:sequence>
        </s:complexType>
      </s:element>
      <s:complexType name="SigningOutput">
        <s:sequence>
          <s:element minOccurs="1" maxOccurs="1" name="signedDocument" type="tns:file" />
          <s:element minOccurs="1" maxOccurs="1" name="certificate" type="tns:binary" />
          <s:element minOccurs="1" maxOccurs="1" name="key" type="tns:binary" />
        </s:sequence>
      </s:complexType>
    </s:schema>
  </wsdl:types>
  <wsdl:message name="SigningServiceSoapIn">
    <wsdl:part name="parameters" element="tns:SigningServiceRequest" />
  </wsdl:message>
  <wsdl:message name="SigningServiceSoapOut">
    <wsdl:part name="parameters" element="tns:SigningServiceResponse" />
  </wsdl:message>
  <wsdl:portType name="SigningServiceSoap">
    <wsdl:operation name="SigningService">
      <documentation xmlns="http://schemas.xmlsoap.org/wsdl/">Signing a document.
      </documentation>
      <wsdl:input message="tns:SigningServiceSoapIn" />
      <wsdl:output message="tns:SigningServiceSoapOut" />
    </wsdl:operation>
  </wsdl:portType>
  <wsdl:binding name="SigningServiceSoap" type="tns:SigningServiceSoap">
    <soap:binding transport="http://schemas.xmlsoap.org/soap/http" style="document" />
    <wsdl:operation name="SigningService">
      <soap:operation soapAction="http://rn.inf.tu-dresden.de:10080/SigningService/
       SigningService" style="document" />
      <wsdl:input>
        <soap:body use="literal" />
      </wsdl:input>
      <wsdl:output>
        <soap:body use="literal" />
      </wsdl:output>
    </wsdl:operation>
  </wsdl:binding>
```

```
  <wsdl:service name="SigningService">
    <documentation xmlns="http://schemas.xmlsoap.org/wsdl/">Signing service.
    </documentation>
    <wsdl:port name="SigningServiceSoap" binding="tns:SigningServiceSoap">
      <soap:address location="http://rn.inf.tu-dresden.de:10080/SigningService.php" />
    </wsdl:port>
  </wsdl:service>
</wsdl:definitions>
```

Listing 5-10: WSDL-Beschreibung des `SigningService`

Auch bei diesem Dienst ergibt sich ein Problem, wenn es sich um einen externen Service Provider handelt. Wie kann ein Dienstanbieter sicher sein, dass das gesendete Dokument auch wirklich vom Absender stammt, für den er das Dokument ja signieren soll. Dieses Problem kann behoben werden, wenn der Dienstnutzer die SOAP-Nachricht, die das zu signierende Dokument enthält, mittels *XML Signature* [W3C01b] selbst signiert. Der Dienstanbieter kann dann sicher sein, dass das Dokument wirklich vom Dienstnutzer gesendet wurde. Dieses Szenario erscheint aber nur sinnvoll, wenn der Dienstanbieter als externer Zeuge den Dokumenttransfer bestätigen soll, ansonsten könnte der Dienstnutzer auch direkt eine signierte SOAP-Nachricht mit dem Dokument an den Empfänger senden. Dieser Dienst scheint deshalb nur innerhalb des Unternehmensnetzwerkes sinnvoll, um für alle Mitarbeiter die Signierung von Dokumenten vorzunehmen. Die Vorteile liegen in diesem Anwendungsfall aber deutlich auf der Hand: Der Dienst kann in vielen Einsatzszenarien wieder verwendet werden, ein einziges Zertifikat für das gesamte Unternehmen ist ausreichend und der geheime Schlüssel zum Signieren muss nur an einer Stelle bekannt sein.

5.2.5.3 Single-Sign-On-Dienst

Ein weiterer sinnvoller Sicherheitsdienst wäre die Verwaltung von Accounts und dazugehörigen Passwörtern verschiedener Dienste oder Anwendungen, um dem Nutzer einen Single-Sign-On zu ermöglichen. Eine ähnliche Intention verfolgt Microsoft mit *.NET Passport*, dessen Nutzung aber bisher in der Fachwelt sehr umstritten ist. Eine einfache Lösung wäre die Verwaltung von Accounts und Passwörtern und den dazugehörigen Diensten mit Hilfe eines Web Service. Der Nutzer kann nur mit einem „Super-Passwort“ auf die hinterlegten Daten zugreifen. Der Vorteil für den Anwender ist, dass er sich nur ein einziges Passwort merken muss. Der Nachteil ist, dass er dieses „Super-Passwort“ entsprechend komplex wählen muss. Tut er dies nicht und können unberechtigte Dritte es brechen, sind natürlich auch alle anderen Accounts unsicher. Um dies zu verhindern, müssen die SOAP-Nachrichten an den Single-Sign-On-Dienst auf jeden Fall mit Hilfe von *XML Encryption* verschlüsselt und am besten zusätzlich mit *XML Signature* unterschrieben werden.

Der Dienst würde folglich als Eingabeparameter den Namen des Nutzers, sein „Super-Passwort“ und den Namen des aufzurufenden Dienstes erwarten. Als Ergebnis werden die Nutzer-ID und das Passwort an den aufzurufenden Dienst übergeben. Weiterhin muss dieser Dienst auch Operationen zum Verwalten der Einträge in der Datenbank zur Verfügung stellen, z.B. `AddAccount` für das Eintragen eines neuen Accounts oder `DelAccount` für das Löschen eines Accounts. Auf die Wiedergabe der ausführlichen Schnittstellenbeschreibung in WSDL wird an dieser Stelle aus Gründen der Übersichtlichkeit verzichtet.

Um eine Überprüfung von Unterschriften durchzuführen, wäre ein weiterer Sicherheitsdienst denkbar. Dieser kann über eine Anfrage bei einer PKI (*Public Key Infrastructure*) erfahren, ob die Unterschrift zu dem Zertifikat des Signierers passt. Um den Zugriff auf PKIs zu standardisieren, wurde von der *W3C XML Key Management Working Group* ein Protokoll namens XKMS (*XML Key Management Specification* [W3C01a]) entwickelt, auf

dessen Basis solche Anfragen an PKIs als Web Services implementiert werden können. Ein Beispiel für einen bekannten XKMS-Dienst ist der von *VeriSign Inc.* - einem bekannten Unternehmen in der Sicherheitsdomäne - unter `http://xkms.verisign.com` bereitgestellte Web Service.

5.2.6 Klassifikationsschema der Telearbeitsbasisdienste

Zusammenfassend lässt sich feststellen, dass eine Vielzahl von verschiedenen Diensten als Web Services abgebildet und somit als Basisdienste zu komplexen Arbeitsabläufen und Geschäftsprozessen orchestriert werden können.

Die aus der Analyse der notwendigen Basisdienste gewonnenen Dienstkategorien wurden in folgendes, in Abbildung 5-7 dargestelltes Klassifikationsschema eingeordnet, welches die Entwickler komplexer Telearbeitsdienste bei der Einordnung der Basisdienste unterstützen soll. Es wurden einige mögliche Basisdienste exemplarisch erwähnt, um die Einordnung zu erleichtern, aber es wird kein Anspruch auf Vollständigkeit erhoben. Die Umsetzung einer Vielzahl von weiteren Diensten mithilfe der Web-Service-Technologie ist denkbar und wird in den nächsten Jahren sicher stetig zunehmen. Anhand dieser Klassifikation wird eine Taxonomie erstellt, mit deren Hilfe die Entwickler die benötigten Basisdienste einordnen und damit die Suche vereinfachen sowie die Wiederverwendung effektiver gestalten können.

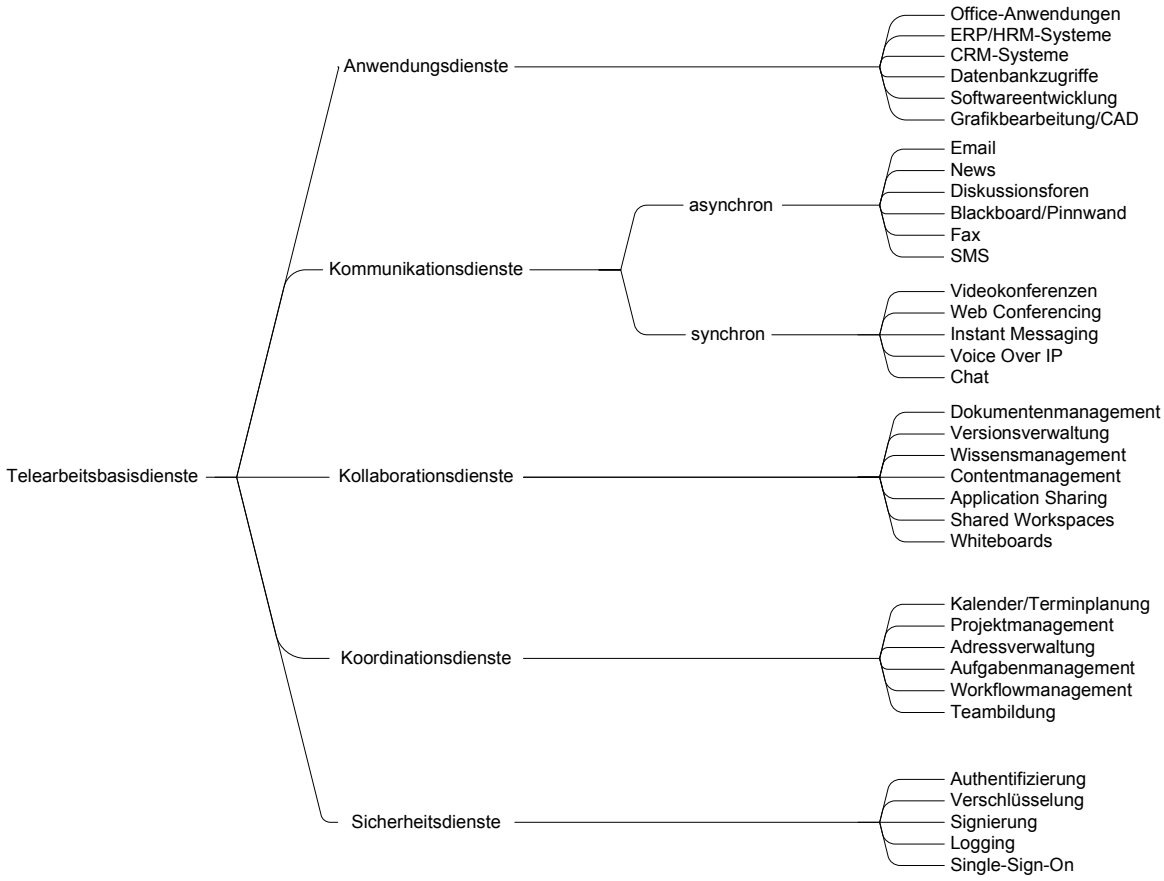

Abbildung 5-7: Klassifikationsschema der Telearbeitsbasisdienste

5.3 Prozessorientierte Integration

Die Integration der verschiedenen Dienste in die Telearbeitsumgebung findet wie bereits beschrieben auf zwei Ebenen statt: auf der Ebene der Benutzer und auf der Ebene der Prozesse. Die benutzerorientierte Integration erfolgt durch ein Portal, welches bestehende für die Benutzerinteraktion aufbereitete Dienste in einer einheitlichen Oberfläche zusammenfasst, die der Nutzer beliebig an seine Wünsche anpassen kann. Die Integration der darunter liegenden Services, die die automatische Zusammenarbeit von Anwendungen im Unternehmen und über Unternehmensgrenzen hinaus möglich macht, folgt dagegen einem prozessorientierten Ansatz, der sowohl die Anforderungen der Softwareintegration als auch die Abbildung der Geschäftsprozesse berücksichtigt. Dem geht eine Strukturierung der vorhandenen Anwendungslandschaft im Unternehmen auf Basis der Modellierung von Anwendungsbausteinen, die entsprechende Services zur Nutzung der angebotenen Funktionalität bereitstellen können, voraus.

In den folgenden Abschnitten wird die Vorgehensweise bei der prozessorientierten Orchestrierung der Telearbeitsdienste detailliert beschrieben. Zur Beschreibung der komplexen Telearbeitsdienste wird die Kompositionssprache WS-BPEL genutzt. Um eine flexible und dynamische Integration der Web Services zu erlauben, wird dem Entwickler ein Framework zur Verfügung gestellt, das die automatische Suche und Auswahl geeigneter Dienste entweder zur Design- oder zur Laufzeit ermöglicht. Dieses Framework wird in den kommenden Abschnitten ebenfalls vorgestellt.

5.3.1 Prozessorientierte Orchestrierung der Basisdienste

Alle für die Erledigung seiner Aufgaben benötigten Dienste kann der Telearbeiter nach Bedarf über den Telearbeitsdienst-Integrator abrufen und in einer einheitlichen Arbeitsumgebung zusammenfügen (siehe Abbildung 5-8). Der Telearbeitsdienst-Integrator übernimmt dabei die Funktion eines Service Brokers, indem er anhand der Suchanfrage des Telearbeiters einen geeigneten Dienst aus einem Pool von Basisdiensten auswählt oder die Suchanfrage an ein öffentliches oder privates Dienstverzeichnis weitersendet. Nach erfolgter Auswahl eines geeigneten Dienstes kann dieser in einen komplexen Ablauf integriert werden. Der entstehende komplexe Telearbeitsdienst initiiert dann entsprechend des vorgegebenen *Process Flows* die Interaktion zwischen Telearbeitsumgebung und dem jeweiligen Diensterbringer.

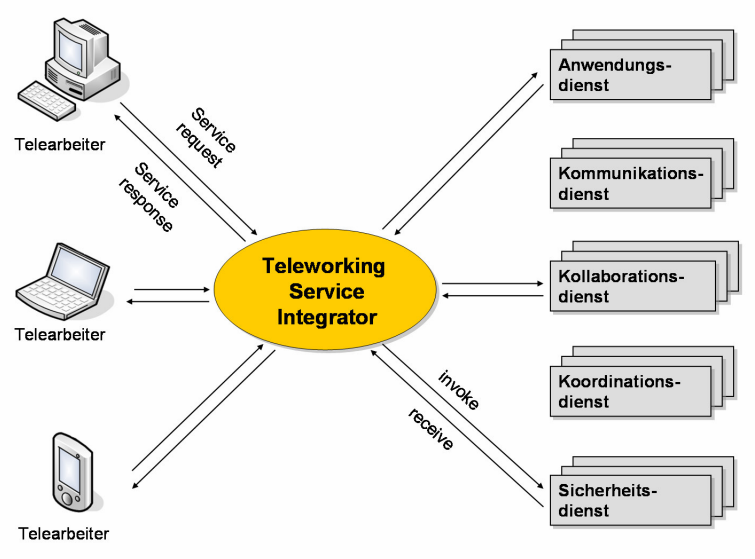

Abbildung 5-8: Orchestrierung der beschriebenen Basisdienste [BrS04]

Das prozessorientierte Vorgehen bei der Integration der benötigten Basisdienste ermöglicht eine einfache Einbeziehung der im Unternehmen ablaufenden Vorgänge. Die Komposition der benötigten Dienste zu Geschäftsprozessen ist dabei unabhängig von der Technik, mit der der Serviceaufruf letztendlich ausgeführt wird. Dabei ist es wichtig, dass die Prozesse möglichst intuitiv modelliert werden können, so dass auch Mitarbeiter die Modelle verstehen und anlegen können, die keine spezielle Ausbildung mitbringen.

Bei der Analyse der Möglichkeiten zur flexiblen Integration der Basisdienste in komplexere Abläufe wurden folgende Arbeitsschritte identifiziert und in einem Workflow zusammengefasst, der in Abbildung 5-9 dargestellt ist.

Als erster Schritt im Entwurfsprozess wird die Abbildung des Geschäftsprozesses oder Arbeitsablaufes auf einen Workflow aus verschiedenen Arbeitsschritten durchgeführt, welcher als abstrakter BPEL-Prozess modelliert wird. Die Verwendung einer abstrakten Prozessbeschreibung erlaubt die Verschiebung der Bindung zu konkreten Web Services auf einen späteren Zeitpunkt. Danach wird festlegt, wie die einzelnen Arbeitsschritte durch Basisdienste abgearbeitet werden können und diese den in Kapitel 5.2 beschriebenen Kategorien von Basisdiensten zugeordnet. Es können auch noch weitere Anreicherungen mit semantischen Informationen vorgenommen werden, um die Suche effektiver zu gestalten.

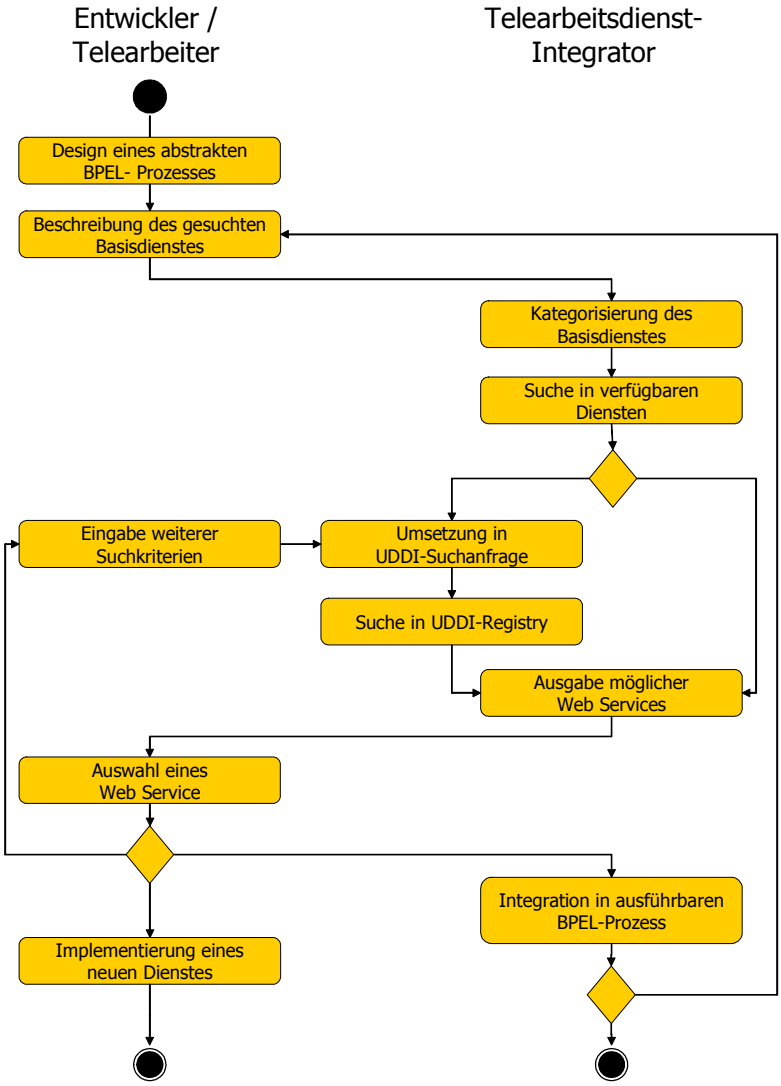

Abbildung 5-9: Workflow zur Dienstintegration

Als Nächstes werden aus einem Pool vorhandener Basisdienste ein oder mehrere passende Dienste ausgesucht und dem Nutzer zur Integration vorgeschlagen. Konnte kein passender Dienst gefunden werden, wird eine Suchanfrage an ein UDDI-Register gestartet. Durch die vorherige Festlegung einschränkender Suchkriterien wird der Suchprozess kürzer und effizienter. Die Beschreibung der Suchkriterien erfolgt dabei mit einem semantischen Modell auf Basis von OWL-S. Kann kein passender Web Service gefunden werden, müssen die formulierten Anforderungen an einen Softwareentwickler weitergeleitet werden, um bedarfsgerecht einen neuen Basisdienst zu implementieren.

Aufbauend auf den vorgestellten Workflow wurde im Ergebnis der vorliegenden Arbeit ein Framework entwickelt, das den Telearbeiter oder einen Entwickler des Unternehmens dabei unterstützt, die verschiedenen Basisdienste einfach in komplexere Abläufe und letztendlich in die Arbeitsumgebung zu integrieren. Dabei soll die Klassifikation der Basisdienste als Grundlage für eine Wiederverwendung bereits vorhandener Dienste und die effektivere Suche nach weiteren Diensten genutzt werden. Ein solches Framework ist in Abbildung 5-10 dargestellt.

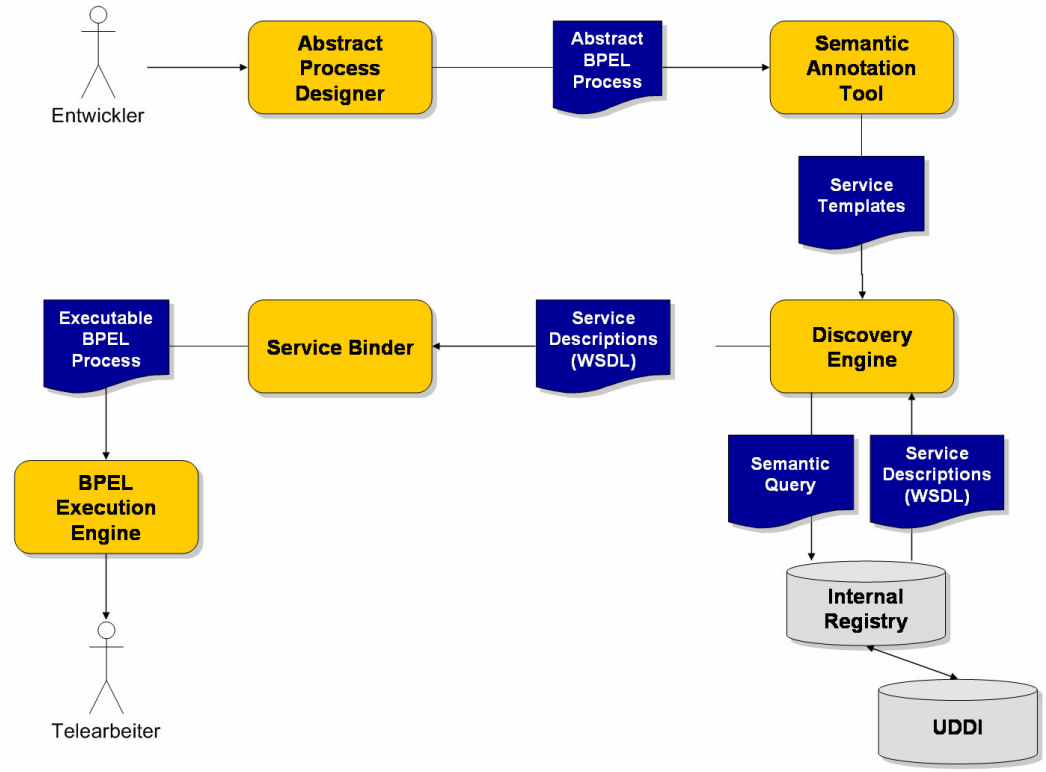

Abbildung 5-10: Framework zur prozessorientierten Integration der Basisdienste

Zur Abbildung der Geschäftsprozesse und Arbeitsabläufe und zur Modellierung eines abstrakten BPEL-Prozesses wird ein *BPEL Process Designer* genutzt, wie z.B. der *BPWS4J Editor* von IBM oder der *BPEL Process Manager* von Oracle. Dabei sollte graphischen Entwicklungstools der Vorzug gewährt werden, um eine intuitive Bedienung auch für unerfahrene Nutzergruppen zu ermöglichen. Mit dem *Semantic Annotation Tool* werden die abstrakten Dienstbeschreibungen mit semantischen Informationen angereichert, wie z.B. der Einordnung in die Basisdienst-Kategorien. Aufbauend auf die annotierten Service-Templates durchsucht die *Discovery Engine* zunächst die interne Registry. Können keine passenden Dienste gefunden werden, wird die Suchanfrage an ein öffentliches UDDI-Register weitergeleitet. Als Ergebnis werden die *Service Descriptions* der nutzbaren Dienste in WSDL an einen *Service Binder* weitergeleitet, der die konkrete Bindung der

Dienste vornimmt und einen *Executable BPEL Process* erzeugt. Dieser kann in einer *BPEL Execution Engine* wie BPWS4J oder dem Oracle *BPEL Process Server* ausgeführt werden und somit vom Telearbeiter wie ein normaler Web Service angesprochen werden.

5.3.2 Suche nach geeigneten Basisdiensten

Wie eben beschrieben werden die abstrakten Dienstbeschreibungen mit semantischen Informationen angereichert, um die Suche zu verbessern und zu beschleunigen. Um die Wiederverwendbarkeit zu erhöhen und die Suche weiter abzukürzen, wird zunächst in einer internen Registry gesucht. Dort sind die semantischen Annotationen bereits hinterlegt, weshalb eine Prüfung der Übereinstimmung zwischen Suchanfrage und vorhandenen Diensten (*Matchmaking*) sehr einfach möglich ist.

Um die vorgenommene Kategorisierung der Basisdienste semantisch zu beschreiben, wird eine Taxonomie erstellt, die Teil einer domainspezifischen Ontologie für den Anwendungsbereich Telearbeit ist. Diese Telearbeitsdienst-Ontologie wird mit Hilfe von OWL modelliert. Ein Ausschnitt aus dieser Beschreibung ist in Listing 5-11 aufgeführt.

```
<rdf:RDF ...>
  <owl:Class rdf:ID="Telearbeitsdienst">
    <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string">komplexer Dienst,
     der aus mehreren Basisdiensten komponiert wird</rdfs:comment>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:someValuesFrom rdf:resource="#TelearbeitsBasisdienst"/>
        <owl:onProperty>
          <owl:ObjectProperty rdf:ID="isComposedOf"/>
        </owl:onProperty>
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
  </owl:Class>
  <owl:Class rdf:about="#Anwendungsdienst">
    <rdfs:subClassOf rdf:resource="#TelearbeitsBasisdienst"/>
  </owl:Class>
  <owl:Class rdf:ID="WS_KundendatenBereitstellen">
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:someValuesFrom>
          <owl:Class rdf:ID="Anwendungsdienst"/>
        </owl:someValuesFrom>
        <owl:onProperty>
          <owl:ObjectProperty rdf:ID="isCategorizedBy"/>
        </owl:onProperty>
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
      <owl:Class rdf:ID="Web Service"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:ObjectProperty rdf:about="#isCategorizedBy">
    <rdfs:domain rdf:resource="#Web Service"/>
    <rdfs:range rdf:resource="#TelearbeitsBasisdienst"/>
  </owl:ObjectProperty>
</rdf:RDF>
```

Listing 5-11: Auszug aus der OWL-Beschreibung der Telearbeitsdienst-Ontologie

In den ersten Zeilen wird beschrieben, dass ein Telearbeitsdienst aus verschiedenen Telearbeitsbasisdiensten zusammengesetzt sein kann (`isComposedOf`). Diese Telearbeitsbasisdienste werden dann in verschiedene Kategorien eingeordnet. Eine Kategorie ist z.B. die der Anwendungsdienste. Weiterhin wird festgelegt, dass ein Web Service zu einer Kategorie von Telearbeitsbasisdiensten zugeordnet werden kann (`isCategorizedBy`). Der Beispieldienst WS_KundendatenBereitstellen wird folglich der Kategorie Anwendungsdienst zugeordnet. Weitere Eigenschaften von Diensten, wie der Dienstanbieter oder

gewünschte Ausgabeparameter, können ebenfalls in OWL beschrieben werden. Dafür kann die standardisierte OWL-S-Ontologie genutzt werden.

Mit einem so genannten OWL-S-*Matchmaker* können dann Suchanfragen, die anhand dieser Ontologie erstellt wurden, mit den Ergebnissen aus dem UDDI-Register verglichen werden. Eine *Mapping Engine* im UDDI-Register verarbeitet die OWL-S-Anfragen, gleicht sie mit den Beschreibungen der veröffentlichten Dienste ab und liefert eine Liste geeigneter Web Services zurück. Grundlage dafür ist natürlich, dass die Dienste bei der Veröffentlichung mit entsprechenden OWL-S-Beschreibungen versehen werden. Eine Infrastruktur zur Umsetzung dieser Suchoperationen ist in Abbildung 5-11 dargestellt.

Weiterhin könnte man auch andere Ontologien bei der semantischen Beschreibung der Dienste berücksichtigen, mit denen man z.B. Quality-of-Service-Eigenschaften beschreiben kann, um die Auswahl der Dienste weiter einzuschränken. So hat bspw. eine Forschergruppe an der Nanyang Technological University in Singapore eine Ontologie namens DAML-QoS entwickelt, die die semantische Beschreibung von QoS-Eigenschaften von Web Services ermöglicht [ZCL05].

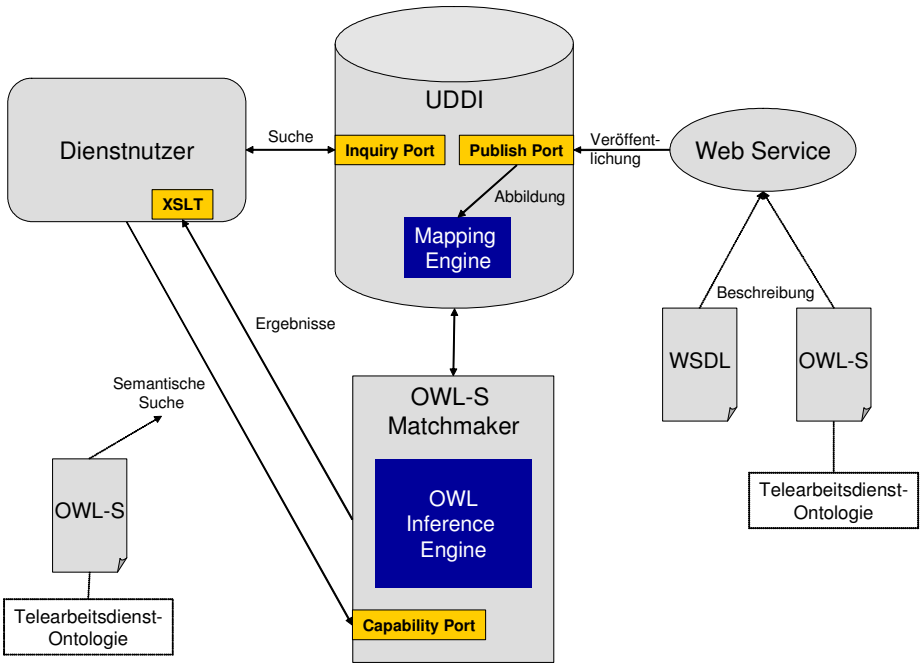

Abbildung 5-11: Infrastruktur für die Verarbeitung semantischer Suchanfragen

Da eine semantische Beschreibung bisher von UDDI-Registern noch nicht unterstützt wird, wurde eine eigene einfache Registry implementiert, die vorhandene Basisdienste mit Hilfe der Telearbeitsdienst-Ontologie beschreibt. Sie enthält außerdem Informationen, ob zu den bekannten Web Services auch GUIDD-Beschreibungen vorhanden sind. Wenn dies der Fall ist, werden diese entsprechend verlinkt. Erst wenn innerhalb dieses eigenen Verzeichnisses keine passenden Basisdienste gefunden werden konnten, wird eine Anfrage an ein öffentliches UDDI-Register erzeugt. Dabei müssen die OWL-S-Beschreibungen auf die UDDI-Einträge abgebildet werden (siehe Abbildung 5-12).

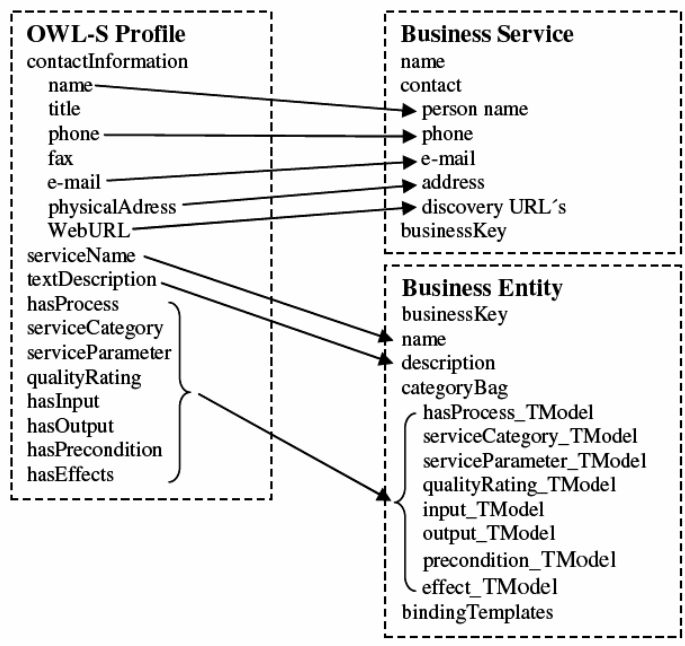

Abbildung 5-12: Mapping von OWL-S-Beschreibungen auf UDDI-Einträge

5.3.3 Komposition der Telearbeitsdienste mittel WS-BPEL

Zur Komposition der Telearbeitsdienste aus den gefundenen Basisdiensten wird die workflowbasierte Orchestrationssprache WS-BPEL benutzt, mit der die Geschäftsprozesse und Arbeitsabläufe, die von einem Telearbeitsdienst abgearbeitet werden sollen, modelliert werden können. Die Funktionsweise dieser Web-Service-Kompositionssprache wurde in Kapitel 4.2.3.2 bereits erläutert und wird im folgenden Kapitel anhand eines Beispiels veranschaulicht.

Um die Bindung an konkrete Basisdienste erst zur Laufzeit durchführen zu können, wird der komplexe Ablauf in BPEL als `abtract process` definiert. Dies ermöglicht die Modellierung eines abstrakten, nicht deterministischen Prozesses, der auf externe Web Services zugreift, ohne deren Ein- und Ausgabe-Parameter vorher zu kennen. Somit können passende Dienste erst zur Laufzeit gesucht werden, was die Flexibilität der erstellten Business-Prozesse und deren Wiederverwendbarkeit verbessert. Um die gewünschte Funktionalität der einzelnen Basisdienste generisch zu beschreiben, werden semantische Annotationen des abstrakten BPEL-Prozesses vorgenommen. Es entstehen so genannte „*Service Templates*“, die an die *Discovery Engine* übergeben werden, um passende Web Services zu finden.

Bei der Komposition verschiedener Basisdienste zu einem Prozess muss nicht nur sichergestellt werden, dass ein angebotener Basisdienst die gewünschte Funktionalität bietet, sondern auch, dass er die von „benachbarten“ Diensten in der Kette zur Verfügung gestellten Daten korrekt interpretieren kann. An der Schnittstelle zwischen zwei Diensten können syntaktische oder semantische „Mismatches“ auftreten, die mit Hilfe von Mediatoren oder Schema-Transformationen behoben werden können.

Syntaktische Heterogenität:

Die gesamte Web-Service-Architektur arbeitet grundlegend auf getypten Daten. Zwei Services können nur Informationen austauschen, wenn diese Informationen einem bestimmten erwarteten Typ entsprechen, folglich syntaktisch kompatibel sind. Liefert der erste Web Service z.B. ein Array von Ausgabeparametern zurück und erwartet der zweite Web Service als Eingabe eine Liste, liegt syntaktische Heterogenität vor.

Im Entwurfsprozess des komplexen Web Services ist folglich zuerst zu klären, ob die beteiligten Dienste syntaktisch kompatibel sind. Syntaktische Kompatibilität von Web Services ist durch einen geeigneten WSDL-Parser leicht zu ermitteln. Das IBM *Emerging Technologies Toolkit (ETTK) for Web Services and Autonomic Computing* [IBM05a] enthält unter anderem einen WSDL-Parser, der zur Überprüfung syntaktischer Kompatibilität verwendet werden könnte. Auch die verschiedenen BPEL-Entwicklungstools unterstützen die Kontrolle syntaktischer Kompatibilität und weisen den Entwickler auf mögliche Datenheterogenitäten hin.

Um syntaktisch heterogene Web Services trotzdem zusammen komponieren zu können, muss eine Transformation der Parameter oder ein Schema-Mapping durchgeführt werden. Zur Transformation können XPATH-Befehle innerhalb von `<assign>` oder eine externe Mapping-Engine eines so genannten Mediators genutzt werden. Dabei wird das Format der Ausgabeparameter des ersten Web Service in das Format der Eingabeparameter des zweiten Web Service transformiert. Die Regeln dieses Mappings werden durch die standardisierte Formatierungssprache XSLT (*eXtensible Stylesheet Language Transformation*) festgelegt, die speziell für die Transformation von XML-Schemata entwickelt wurde.

Semantische Heterogenität:

Semantische Heterogenität liegt z.B. vor, wenn der erste Web Service einen Parameter `Name` zurückliefert, der als Wert den Vornamen einer Person enthält. Der zweite Web Service erwartet als Eingabe für den Parameter `Name` jedoch den Nachnamen der Person. In diesem Fall würden beim einfachen Kopieren der Werte innerhalb von `<assign>` keine syntaktischen Fehler auftreten, aber das Ergebnis des zweiten Web Service wäre falsch. Ohne ein Schema-Mapping mittels XSLT könnten die beiden Dienste nicht automatisch zusammenarbeiten.

Eine Möglichkeit, semantische Heterogenität zu erkennen oder auch zu vermeiden, ist die Nutzung von WSDL-S [IBM05a]. Mit WSDL-S kann der Entwickler semantische Annotationen zu einem WSDL-Dokument hinzufügen, welche die Semantik der Input- und Output-Parameter, der Vorbedingungen und Ergebnisse der Operationen beschreiben. Diese werden mit Ontologien für einen spezifischen Anwendungsbereich (*Domain Model*) verknüpft. Diese Ontologien werden z.B. unter Verwendung von OWL (*Web Ontology Language*) spezifiziert. Ein so genannter *Matchmaker* vergleicht dann die Beschreibung des Dienstes mit der Ontologie und erkennt, ob semantische Heterogenität oder eine Übereinstimmung der Semantik vorliegt. Wird die semantische Beschreibung bereits in die Suchanfrage integriert, kann man von vornherein ausschließen, dass semantisch heterogene Basisdienste angeboten werden.

Entsprechende Tools sind bereits auf dem Markt erhältlich. IBM z.B. bietet als Teil des *ETTK for Web Services and Autonomic Computing* [IBM05a] Eclipse Plug-Ins für semantische Annotationen innerhalb der WSDL-Beschreibung (*Semantic Annotation Editor*) und für die Überprüfung der semantischen Kompatibilität von Web Services (*Semantic Annotation Matchmaker*) an.

5.3.4 Beispiel für einen komplexen Telearbeitsdienst

Im folgenden Abschnitt soll die Vorgehensweise bei der Erstellung eines komplexen Telearbeitsdienstes am Beispiel eines Dienstes zum Abschluss eines Werbevertrages durch einen Telearbeiter veranschaulicht werden. Der Telearbeiter ist Angestellter einer Werbeagentur, die mit einem Kunden einen Werbevertrag abschließen möchte. Als erstes ruft der Telearbeiter über einen Web Service, der eine Schnittstelle zum CRMS[53] seines Unternehmens herstellt, die Daten des Kunden ab. Um einen konkreten Ansprechpartner beim Kunden zu identifizieren, startet er nun eine Anfrage an einen Dienst des Kunden, der die Kontaktdaten und Verantwortlichkeiten der Mitarbeiter zurückliefert. Das Ergebnis wird als Input an einen Dienst zur Kommunikation mit dem Verantwortlichen weitergegeben. Dieser Kommunikationsdienst kann entweder ein E-Mail-Dienst oder, wenn die Reaktionszeit verringert werden soll, ein Conferencing-Dienst zur synchronen Kommunikation sein. Hat er die Einzelheiten mit dem Kunden besprochen und liegen ihm die Informationen aus dem Produktkatalog des Kunden vor, erstellt der Telearbeiter ein Angebot und sendet es an den Kunden zur Prüfung. Da es sich um ein vertrauliches Dokument handelt, wird es zuvor durch einen Sicherheitsdienst verschlüsselt und dann erst versendet. Wenn dieser eine positive Rückmeldung gibt, kann der Vertrag unterzeichnet werden. Wenn nicht, wird eine erneute Kommunikation mit dem Kunden angestoßen und die weiteren Schritte wiederholt.

Der Ablauf dieses Workflows ist in Abbildung 5-13 detailliert dargestellt. Die einzelnen Dienste stellen die Beschreibung ihrer Schnittstellen mittels WSDL bereit und können so in den komplexen Ablauf eingebunden werden.

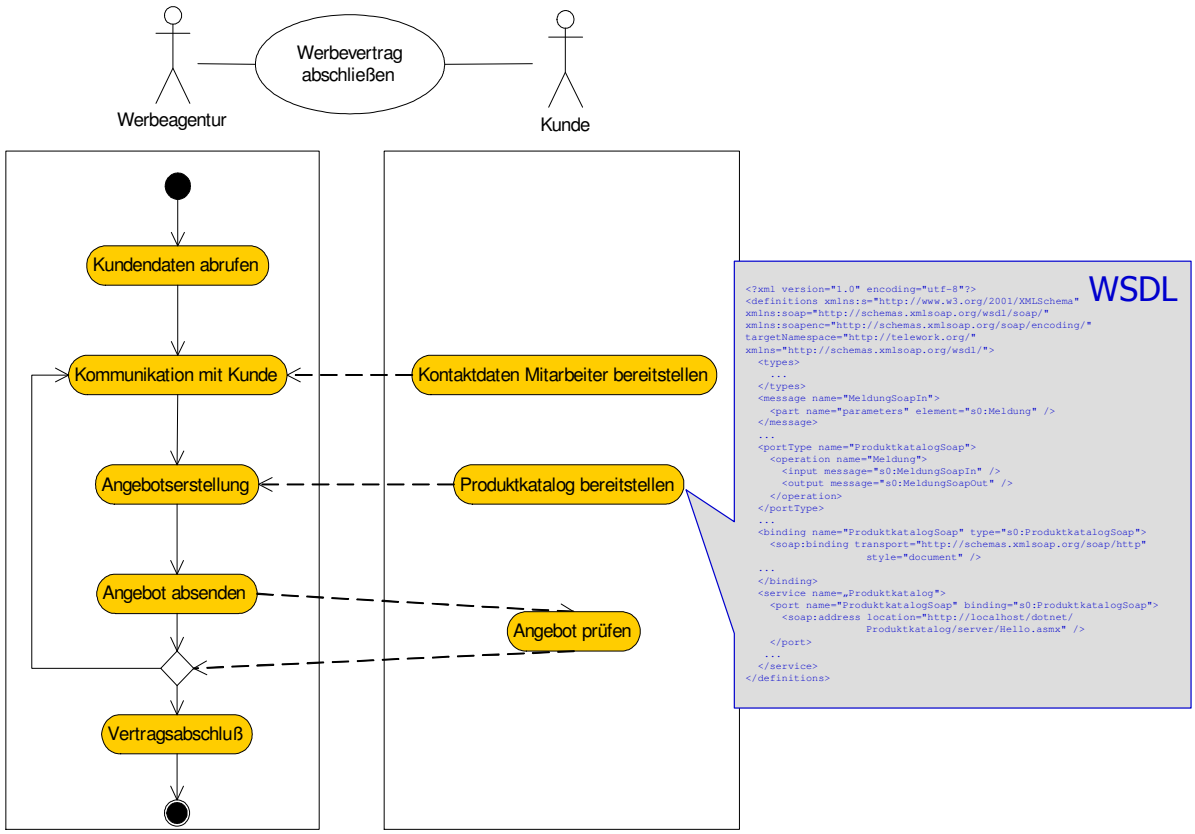

Abbildung 5-13: Beispiel für einen komplexen Workflow

[53] CRMS (Customer Relationship Management System) sind betriebswirtschaftliche Software-Lösungen zur Verwaltung von Kunden und deren Beziehungen zum Unternehmen.

Um die benötigten Dienste effizienter zu suchen oder aus einem Pool bereits verfügbarer Dienste aussuchen zu können, werden sie einer Kategorie von Basisdiensten laut der Klassifizierung in Kapitel 5.2 zugeordnet (siehe Tabelle 5-1).

Dienst	Kategorie von Basisdiensten
KundendatenBereitstellen	Anwendungsdienst
KontaktDatenMitarbeiterBereitstellen	Anwendungsdienst
KommunikationKunde	Kommunikationsdienst
ProduktkatalogBereitstellen	Anwendungsdienst
AngebotErstellen	Anwendungsdienst
DokumentVerschlüsseln	Sicherheitsdienst
AngebotAbsenden	Kollaborationsdienst/ Kommunikationsdienst
AngebotPrüfen	Anwendungsdienst
Vertragsabschluss	Anwendungsdienst

Tabelle 5-1: Kategorisierung der benötigten Basisdienste

Um die Suche weiter einzuschränken, müssen nun weitere Eigenschaften der benötigten Dienste auf Basis der Telearbeitsdienst-Ontologie beschrieben werden.

`KundendatenBereiststellen`: Mit diesem Dienst soll ein Zugriff auf das CRM-System des eigenen Unternehmens stattfinden. Anbieter des Service ist folglich auf jeden Fall das eigene Unternehmen (im Beispiel Teleworx Inc.). Weiterhin kann als Quelle der Daten ein *Customer Relationship Management System* (CRMS) identifiziert werden. Dieses Wissen kann die Suche nach einem passenden Dienst deutlich einschränken. In RDF können diese Eigenschaften wie folgt abgebildet werden (siehe Abbildung 5-14):

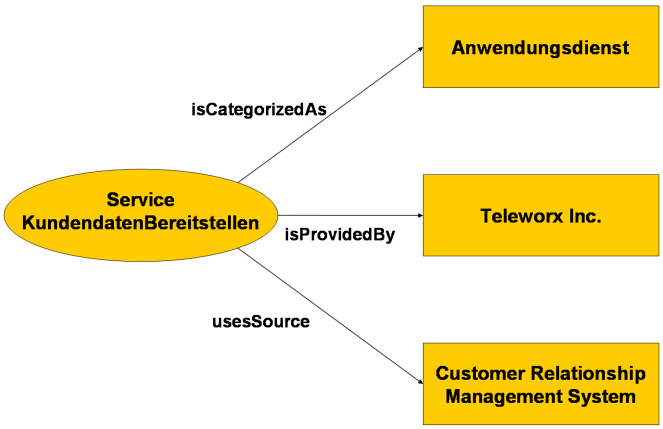

Abbildung 5-14: RDF-Beschreibung der Eigenschaften des Dienstes `KundendatenBereitstellen`

`KontaktdatenMABereitstellen`: Dieser Anwendungsdienst soll die Kontaktdaten des verantwortlichen Mitarbeiters im Unternehmen des Kunden bereitstellen. Es erfolgt also ein Zugriff auf das Personalverwaltungssystem des Kunden. Der Dienst wird folglich vom Kunden (im Beispiel Test AG) bereitgestellt und nutzt als Quelle der Informationen ein *Human Resource Management System* (HRMS). Die RDF-Darstellung ist in Abbildung 5-15 zu sehen.

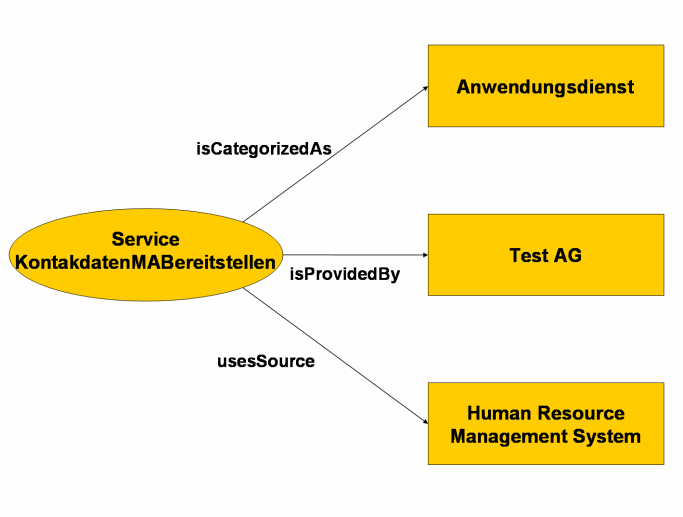

Abbildung 5-15: RDF-Beschreibung des Dienstes `KontaktdatenMABereitstellen`

`KommunikationKunde`: Hier soll ein Dienst zur Kommunikation mit dem Kunden genutzt werden. Dieser kann aus einem Pool bereits vorhandener Basisdienste ausgewählt werden, was die Wiederverwendbarkeit der Dienste und damit die Effizienz der Lösung erhöht. Dabei ist grundsätzlich offen, ob ein synchroner oder asynchroner Kommunikationsdienst gewünscht wird. Dieses Kriterium könnte als einschränkende Bedingung vom Telearbeiter vorgegeben werden, wenn die Auswahl an verfügbaren Diensten zu umfangreich ist. Befindet sich der Kommunikationspartner z.B. in einem Land mit einer anderen Zeitzone, ist eine synchrone Kommunikation sicher nicht so zweckmäßig, da die Arbeitszeiten entsprechend differieren. Es wäre aber auch denkbar, dass der Kunde, mit dem kommuniziert werden soll, vorgibt, welche Kommunikationsdienste er überhaupt nutzen kann. Ist sein Arbeitsplatz nicht mit einer Videokamera und einem Headset oder Mikrofon ausgestattet, ist eine Nutzung von Video- oder Audiokonferenzlösungen nicht möglich. Grundsätzlich wäre hier auch die Nutzung mehrerer verschiedener Kommunikationsdienste denkbar. Diese können dann in alternativen Ausführungspfaden (`switch`) im komplexen Web Service abgebildet werden. Der Telearbeiter kann dann erst zur Laufzeit des Dienstes einen bestimmten Kommunikationsdienst auswählen.

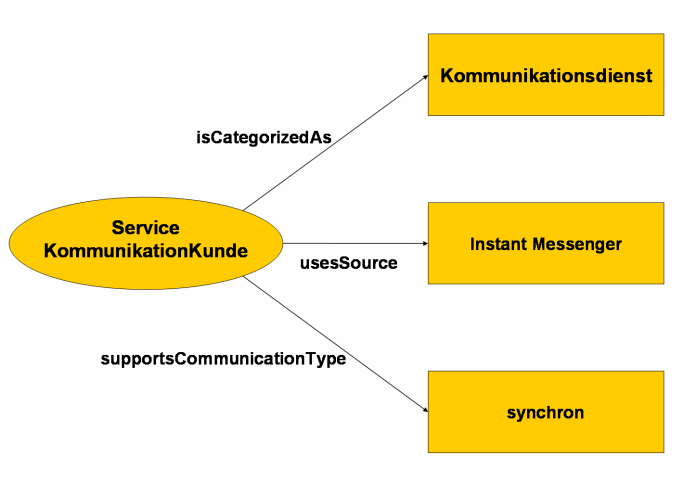

Abbildung 5-16: RDF-Beschreibung des Dienstes `KommunikationKunde`

`ProduktkatalogBereitstellen`: ist ebenfalls ein Anwendungsdienst, der vom Kunden angeboten wird und auf dessen ERP-Systeme zugreift.

`AngebotErstellen`: Dies ist ein Anwendungsdienst, den der Telearbeiter nutzt, um aus den bisher erhaltenen Informationen ein Angebot für den Kunden zu erstellen. Hier ist es möglich, dass ein solcher Dienst bereits im Unternehmen des Telearbeiters genutzt wird und damit wieder verwendet werden kann. Andererseits könnte dieser Dienst auch im Rahmen von Outsourcing durch ein drittes Unternehmen bereitgestellt werden. Bei der Auswahl des richtigen Anbieters können dann Kriterien wie der Preis oder die Einhaltung von Quality-of-Service-Parametern eine Rolle spielen.

`DokumentVerschlüsseln`: Da das Angebot ein vertrauliches Dokument ist, wird es vor der Versendung an den Kunden verschlüsselt. Dieser Sicherheitsdienst sollte entweder vom eigenen Unternehmen oder einer vertrauenswürdigen dritten Instanz erbracht werden. Sollte es eventuell zu Unstimmigkeiten zwischen den Geschäftspartnern kommen, könnte diese Instanz auch als Zeuge auftreten. Eingabeparameter für den Dienst sind das Dokument, das bevorzugte Verschlüsselungsverfahren und die Schlüssellänge. Weitere Details der Schnittstellenbeschreibung und Implementierung können in Kapitel 5.2.5.1 nachgelesen werden.

`AngebotAbsenden`: Die Aufgabe dieses Dienstes, dem Kunden das Angebot zur Verfügung zu stellen, kann auf unterschiedliche Weise erfüllt werden. Hier wäre sowohl die Nutzung von Kommunikations- als auch von Kollaborationsdiensten denkbar. Der Telearbeiter könnte das Angebot z.B. per E-Mail an den Kunden senden, also einen Kommunikationsdienst nutzen. Es wäre aber auch möglich, dass er das Angebot auf einem so genannten *Shared Workspace* ablegt, auf den der Kunde zugreifen kann. Dieser Service wird als Kollaborationsdienst kategorisiert. Auch hier wäre wieder eine parallele Integration mehrerer Dienste denkbar, die dann erst zur Laufzeit vom Telearbeiter ausgewählt werden. Dadurch würde die Abbildung des komplexen Ablaufes generischer und damit besser in anderen Anwendungsszenarien wiederverwendbar.

`AngebotPrüfen`: Dieser Dienst wird von Seiten des Kunden angeboten. Bei der Integration des Dienstes spielt es erst einmal keine Rolle, ob die Prüfung automatisiert oder manuell beim Kunden erfolgt. Für den Telearbeiter ist es nur von Bedeutung, als Ausgabeparameter eine Nachricht zu erhalten, ob das Angebot so akzeptiert wird oder nicht. Wie diese Nachricht beim Kunden erzeugt wird, bleibt für den Telearbeiter weitestgehend transparent. Sollte der Kunde das Angebot nicht akzeptieren, wird eine erneute Kommunikation mit ihm angestoßen und der Vereinbarungsprozess erneut durchlaufen. Dies kann in WS-BPEL durch einen bedingten Rücksprung (`switch`) oder eine kopfgesteuerte Schleife (`while`) abgebildet werden.

`Vertragsabschluss`: Der Dienst `Vertragsabschluss` ist wiederum ein Anwendungsdienst auf Seiten des Telearbeiters. Er kann verschiedene Funktionalitäten kapseln, also selbst ein komplexer Web Service sein. Dies spielt für die Integration in den Geschäftsprozess aber keine Rolle. Ein solcher Dienst kann z.B. von der Rechtsabteilung des eigenen Unternehmens des Telearbeiters zur Verfügung gestellt werden, um die für das Unternehmen spezifischen Anforderungen und Standards in die Vertragsbedingungen einfließen zu lassen.

In WS-BPEL kann die Komposition des komplexen Web Service `WerbevertragAbschließen` wie folgt abgebildet werden (Abbildung 5-17):

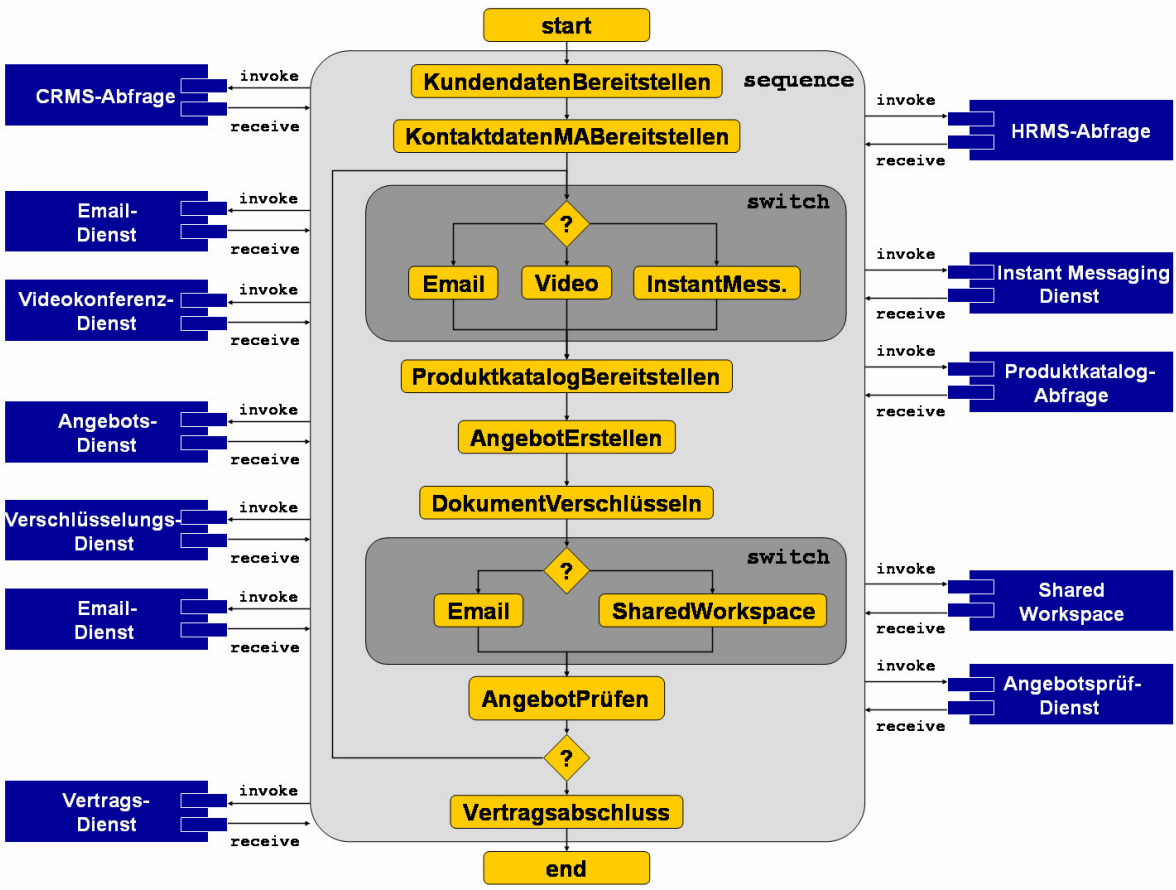

Abbildung 5-17: BPEL Process Flow des komplexen Web Service WerbevertragAbschliessen

Ein Auszug aus dem BPEL-Quellcode des komplexen Web Service *WerbevertragAbschliessen* ist in Listing 5-12 zu sehen.

```
<process name="WerbevertragAbschliessen" …>
  <partnerLinks>
    <!-- Die 'client' Rolle repräsentiert den Service Consumer, 'myRole' definiert die
     Rolle des BPEL-Prozesses -->
    <partnerLink name="client" partnerLinkType="tns:WerbevertagAbschliessen"
     myRole="WerbevertagAbschliessenProvider"
     partnerRole="WerbevertragAbschliessenCustomer"/>
    <partnerLink name="KundendatenBereitstellen"
     partnerLinkType="tns:KundendatenBereitstellen" partnerRole="KundendatenProvider"/>
    <partnerLink name="KontaktdatenMABereitstellen"
     partnerLinkType="tns:KontaktdatenVerantwortliche"
     partnerRole="KontaktdatenVerantwortlicheProvider"/>
    <partnerLink name="KommunikationsDienst" partnerLinkType="tns:KommunikationsDienst"
     partnerRole="KommunikationsDienstProvider"/>
    <partnerLink name="Produktkatalog" partnerLinkType="tns:Produktkatalog"
     partnerRole="ProduktkatalogProvider"/>
    <partnerLink name="AngebotErstellen" partnerLinkType="tns:AngebotErstellen"
     partnerRole="AngebotErstellenProvider"/>
    <partnerLink name="DokumentVerschlüsseln" partnerLinkType="tns:DokumentVerschlüsseln"
     partnerRole="DokumentVerschlüsselnProvider"/>
    <partnerLink name="AngebotSenden" partnerLinkType="tns:AngebotSenden"
     partnerRole="AngebotSendenProvider"/>
    <partnerLink name="AngebotPrüfen" partnerLinkType="tns:AngebotPrüfen"
     partnerRole="AngebotPrüfenProvider"/>
    <partnerLink name="Vertragsabschluss" partnerLinkType="tns:Vertragsabschluss"
     PartnerRole="VertragsabschlussProvider"/>
  </partnerLinks>

  <variables>
       <!--   Deklaration der Variablen -->
  </variables>
```

```
  <sequence>
    <receive partnerLink="client" portType="tns:WerbevertragAbschliessenPT"
     operation="WerbevertragAbschliessen" variable="WerbevertragAbschliessenRequest"
     createInstance="yes"/>

    <assign>
      <!-- Zuweisung der Variablen -->
      <copy>
        <from variable="WerbevertragAbschliessenRequest" part="Kunde"/>
        <to variable="KundendatenBereitstellenRequest" part="Kundenname"/>
      </copy>
    </assign>

    <invoke partnerLink="KundendatenBereitstellen"
     portType="tns:KundendatenBereistellenPT"
     operation="KundendatenBereitstellen" inputVariable="KundendatenBereitstellenRequest"
     outputVariable="KundendatenBereitstellenResponse" />

    <assign>
     ...
    </assign>

    <invoke ... />
      ...
    <switch>
      <case condition ...> ... </case>
      <otherwise> ... </otherwise>
    </switch>

    <invoke partnerLink="client" portType="tns:ClientCallbackPT"
     operation="ClientCallback"
     inputVariable="WerbevertragAbschliessenResponse"/>

</sequence>

</process>
```

Listing 5-12: Auszug aus dem WS-BPEL-Quellcode des komplexen Web Service `WerbevertragAbschliessen`

Das Element `<partnerLink>` enthält alle Verbindungsinformationen zu den Web Services, die im vorliegenden komplexen Web Service genutzt werden. Dabei wird durch die Attribute `partnerLinkType` und `partnerRole` zum einen genau festlegt, mit welchem Web Service eine Zusammenarbeit stattfinden soll, und zum anderen, welche Funktion dieser im komplexen Ablauf einnimmt.

Mit dem `<sequence>`-Element wird der sequentielle Ablauf der einzelnen Service-Aufrufe dargestellt. Als erstes wird immer die Entgegennahme des Dienstaufrufes durch den Client durchgeführt, was durch das Element `<receive partnerLink="client">` beschrieben wird. Mit `<assign>` wird die Übergabe der Werte zwischen den Outputvariablen des vorhergehenden Prozesses und den Inputvariablen des folgenden Prozesses geregelt. Zum Aufruf eines Web Service wird dann `<invoke>` genutzt, wobei der `portType`, die Operation und die Input- und Outputvariablen definiert werden. Will man alternative Verläufe in Abhängigkeit von einer Bedingung darstellen, nutzt man das `<switch>`-Tag. Es ermöglicht die Angabe einer Bedingung mit dem `<case condition>`-Element.

5.3.5 Fazit

WS-BPEL bietet zur Beschreibung der Abläufe und Abhängigkeiten sowohl statische als auch abstrakte Prozessmodelle an. Somit kann die Bindung konkreter Basisdienste entweder durch den Entwickler zur Designzeit oder durch den *Teleworking Service Integrator* zur Laufzeit erfolgen.

Meist sind die Arbeitsabläufe fest vorgegeben und hinlänglich bekannt, so dass sie durch einen Entwickler vorab komponiert werden können. Diese Komposition erfolgt meist

durch die IT-Abteilung des Unternehmens, in dem der Telearbeiter beschäftigt ist, welches dann selbst als Service Provider gegenüber dem Telearbeiter auftritt. Die Auswahl der genutzten Web Services kann durch das entwickelte Framework dahingehend dynamisch gestaltet werden, dass zur Designzeit nur abstrakte Schnittstellen-Beschreibungen definiert werden und erst zur Laufzeit durch den *Teleworking Service Integrator* aus einem Pool von Web Services mit gleicher Funktionsweise ein spezieller Dienst ausgewählt wird. Dies setzt aber voraus, dass Kriterien zur automatischen Auswahl, wie z.B. Quality-of-Service-Eigenschaften, vorher festgelegt werden.

Vorteil der Komposition der einzelnen Basisdienste zu einem komplexen Telearbeitsdienst ist, dass der Telearbeiter den kompletten Ablauf als einen Dienst ansprechen und ihn so in das Portal integrieren kann. Grundsätzlich muss er sich nicht mit der dahinter liegenden Implementierung der einzelnen Dienste und deren Zusammenspiel auseinandersetzen, um diesen Arbeitsschritt effizient durchzuführen. Mit dem beschriebenen Beispiel konnte gezeigt werden, wie die Abbildung verschiedener Arbeitsabläufe eines Telearbeiters auf komplexe Web Services erfolgen kann.

5.4 Benutzerorientierte Integration

Ziel ist es, alle benötigten Dienste in einer gemeinsamen Benutzeroberfläche zusammenzuführen. Die Erstellung einer anwendungsübergreifenden Oberfläche sollte sich dabei stark an den Arbeitsabläufen der Telearbeiter und nicht an den Applikationsgrenzen orientieren. Der Benutzer soll das Gefühl haben, er arbeite mit einer einzigen Applikation, die genau für seine Bedürfnisse optimiert wurde.

5.4.1 Schaffung einer einheitlichen Benutzeroberfläche mittels Portaltechnologie

Wie bereits in Kapitel 5.1.2 beschrieben soll die benutzerorientierte Integration der Telearbeitsdienste in eine einheitliche Umgebung losgelöst von der eigentlichen Anwendungslogik in der obersten Ebene der Architektur stattfinden. Zur Schaffung einer einheitlichen Benutzeroberfläche für alle Telearbeitsdienste wird dabei die Portaltechnologie genutzt. Portale ermöglichen den Zugang über einen Webbrowser und erfüllen damit die Forderung nach einem Thin Client. Die Webseiten werden dabei in Portlets unterteilt, die vom Portalserver erzeugt und verwaltet werden. Diese Portlets können flexibel angeordnet und angepasst werden. Außerdem bieten Portale eine umfangreiche Sicherheitsunterstützung wie Nutzer- und Rollenverwaltung oder Single-Sign-On sowie Werkzeuge zur Personalisierung und Adaption an verschiedene Endgeräte. Weitere Gründe für die Auswahl dieser Technologie zur benutzerorientierten Integration der Telearbeitsdienste wurden bereits in Kapitel 5.1.4 dargelegt.

Das über einen Webbrowser zugängliche Portal führt die einzelnen Dienste für die Anwender zu einem einheitlichen System zusammen und bietet ihnen einen rollenbasierten, transparenten Zugriff auf alle Informationen und Anwendungen, die sie zur Erledigung ihrer Aufgaben benötigen, ohne die Quelle der Informationen oder die Backend-Systeme kennen zu müssen (siehe Abbildung 5-18).

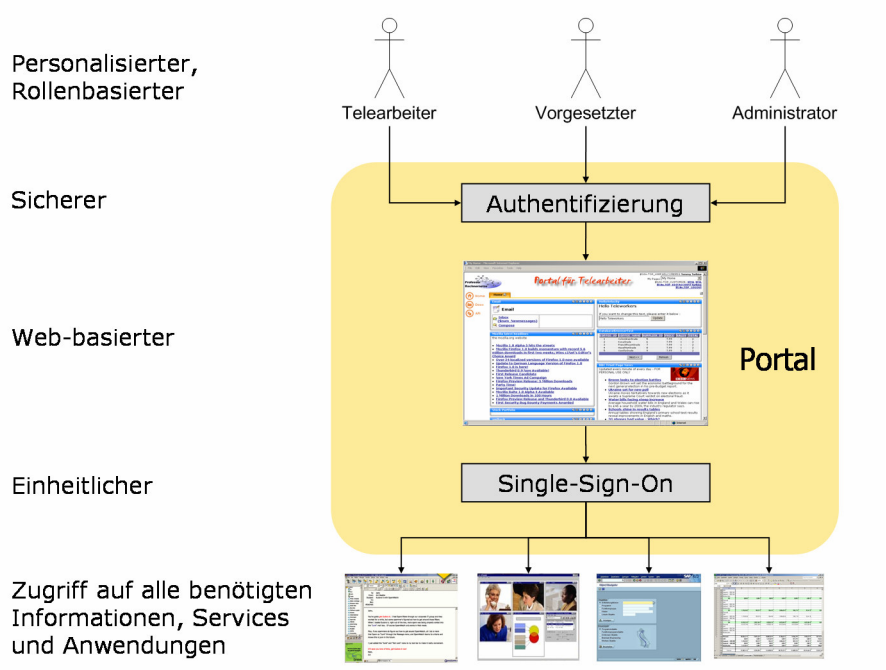

Abbildung 5-18: Benutzerorientierte Integration durch Nutzung der Portaltechnologie

5.4.2 Flexible Dienstintegration

Grundsätzlich kann jede Applikation in einem Browser gestartet werden und damit auch in ein Portal integriert werden. Tatsache ist aber, dass mit der Komplexität der Anwendung auch die Schwierigkeit der Einbindung in das Portal steigt. Kritisch ist z.B. die Einbindung von Anwendungen, die nur mit sehr komplexen Oberflächenfunktionalitäten bedient werden können, wie CAD- oder Grafikprogramme. Außerdem müssen in Portalen normalerweise für jede Anwendung passende Portlets implementiert werden, was eine flexible Integration neuer Dienste erschwert.

Um die Telearbeitsdienste flexibel in das Portal integrieren zu können, muss zuerst ein Portlet bereitgestellt werden, dass als Container für den Web-Service-Aufruf fungiert. Viele Portalhersteller bieten vordefinierte Templates für Web-Service-Portlets an. Diese werden aber an einen konkreten Web Service gebunden, in dem die WSDL-Schnittstelle des Dienstes verlinkt wird. Um eine flexible Dienstintegration zu ermöglichen, muss das Portlet aber erst zur Laufzeit die Schnittstellenbeschreibung des Dienstes abfragen und eine an den Dienst angepasste GUI anzeigen. Web Services haben aber eigentlich keine eigene Benutzeroberfläche, da sie originär für die Kommunikation mit anderen Anwendungen und nicht mit Endnutzern gedacht waren. Um sie aber aus einem Portlet heraus aufrufen zu können, müssen die Eingabeparameter über ein Formular an den Dienst übergeben werden. Wie kann man ein solches Formular nun zur Laufzeit generieren?

Die Schnittstellen eines Web Service werden in WSDL spezifiziert, um den genauen Datenaustausch und die verwendeten Datentypen festzulegen. Daher liegt es nahe, dieses ohnehin schon vorhandene Dokument zur Entwicklung einer Benutzeroberfläche für Web Services heranzuziehen. Das WSDL-Dokument beinhaltet alle Daten, die der Anwender mit dem Web-Service-Anbieter austauscht und zusätzlich auch das Format der Datentypen. Da WSDL auf XML aufbaut, kann somit das WSDL-Dokument geparst werden, um so generisch eine GUI aufzubauen. Damit die Formularfelder nicht mit den meist kryptischen Parameternamen versehen werden, ist es sinnvoll, zusätzliche Beschreibungen der GUI-Elemente vorzunehmen und diese bei der Erzeugung des HTML-Formulars zu berücksich-

tigen. Entsprechende Ansätze wurden bereits in Kapitel 4.2.5 ausführlich beschrieben. Weiterhin können damit auch Hilfetexte zu den Formularfeldern und mehrseitige Formulare erzeugt werden.

Der Workflow der Integration eines Telearbeitsdienstes in das Portal ist in Abbildung 5-19 übersichtlich dargestellt.

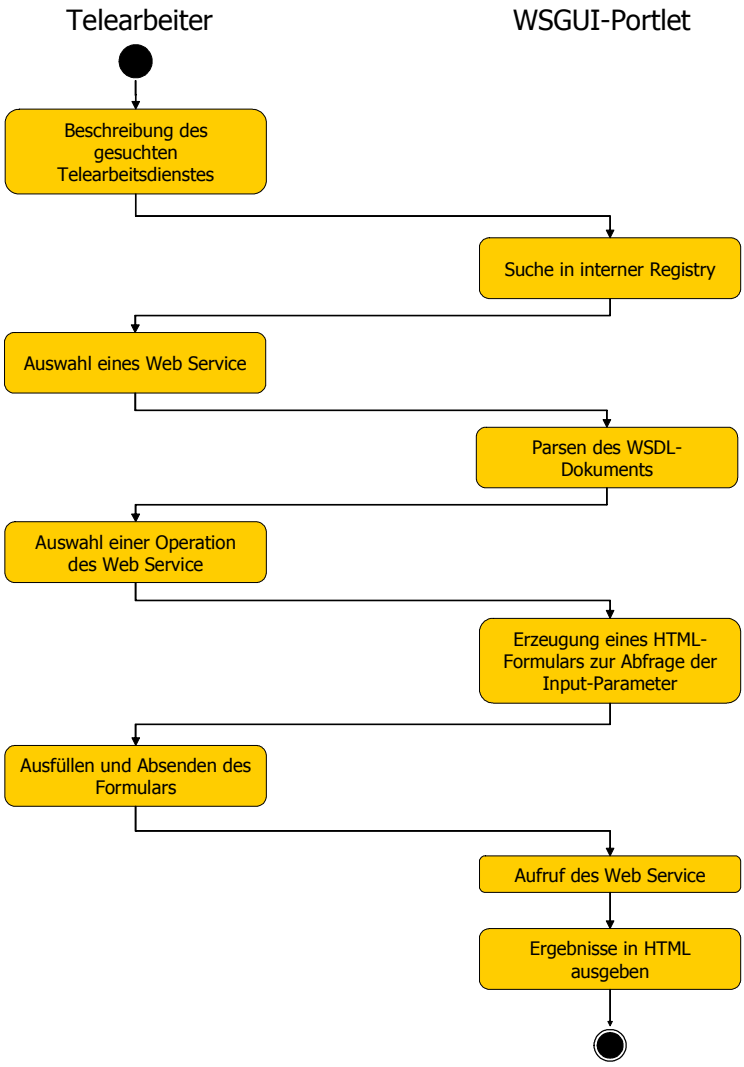

Abbildung 5-19: Workflow der Dienstintegration in das Portal

Dabei ist die Beschränkung auf die Suche in einer internen Registry bewusst gewählt, damit der Telearbeiter nur Dienste zur Integration auswählen kann, die vorher von der IT-Abteilung des Unternehmens auf ihre Funktionsweise und Sicherheit getestet wurden. Andernfalls würde die Integration beliebiger Web Services aus unbekannten Quellen ein erhebliches Sicherheitsrisiko darstellen. Außerdem ermöglicht es den Entwicklern, zu den registrierten Web Services auch entsprechende GUI-Beschreibungen in der Registry abzulegen.

Besonderheit bei der Verarbeitung von Web Services mit mehreren Operationen ist die Auswahl einer einzelnen Operation, bevor das HTML-Formular erzeugt wird, da sich die Eingabeparameter der Operationen normalerweise unterscheiden. Genauere Details zur Implementierung der Integration werden in Kapitel 6 beschrieben.

5.4.3 Personalisierung und Individualisierung

Damit die Telearbeiter ihre Aufgaben noch effizienter erledigen können, sollen sie die Möglichkeit haben, ihre Arbeitsumgebung individuell anzupassen. Hierzu können sie Oberflächenelemente, so genannte Portlets (siehe Kapitel 4.3.2), beliebig anordnen, deren Farben und Formen (*look and feel*) bestimmen und damit die Oberflächenergonomie optimal auf ihre persönlichen Arbeitsabläufe abstimmen. Neben den verschiedenen Anordnungs- und Darstellungsmöglichkeiten können Portlets auch ausgeblendet oder inhaltlich verändert werden.

Nach einer erfolgreichen Authentifizierung generiert der Portalserver die Portalseiten dynamisch entsprechend dem gespeicherten Profil des Telearbeiters. Diese Profildaten enthalten unter anderem individuelle Farb- und Navigationseinstellungen, Rechte für einzelne Dienste und eine Zusammenstellung anwender- bzw. rollenspezifischer Dienstquellen. Benutzerprofile können auf unterschiedliche Weise konfiguriert werden. Dabei ist generell zwischen Personalisierung und Individualisierung zu unterscheiden. Konfiguriert der Benutzer seine Einstellungen selbst, spricht man von Individualisierung. Im Falle der Personalisierung wird die Konfiguration durch das System oder einen Administrator vorgenommen. In den meisten Fällen wird die Grundkonfiguration vom System oder dem Administrator über die Zuteilung einer Rolle personalisiert. Der Nutzer hat dann die Möglichkeit, sein Profil entsprechend seinen Bedürfnissen zu individualisieren.

Individualisierung birgt aber auch eine Reihe von Gefahren. Bei einer zu starken Individualisierung ist ein Support-Mitarbeiter beispielsweise nicht mehr in der Lage, Soforthilfe bei Problemen mit der Benutzeroberfläche zu leisten, da er die genaue Konfiguration auf Seiten des Telearbeiters gar nicht kennt. Individualisierung sollte deshalb nur beschränkt zugelassen werden und die Konfiguration anhand der Rollen vorgegeben werden. Durch eine entsprechende Hierarchisierung der Rollen und Nutzer ist eine Abstufung der Berechtigungen zur Individualisierung möglich. Entsprechend geschulten Mitarbeitern können so höhere Freiheitsgrade bei der Individualisierung eingeräumt werden als jenen, die mit dem System weniger vertraut sind.

5.4.4 Adaptierbarkeit

Portale bieten standardmäßig nicht nur den Zugang über einen Webbrowser an, sondern unterstützen auch die Nutzung anderer Client-Programme und Endgeräte. So wird von den meisten Herstellern eine WAP-Schnittstelle oder die Unterstützung von VoiceXML angeboten. Vor allem für mobile Telearbeiter ist die Nutzung verschiedener Endgeräte sehr wichtig, da sie nicht immer einen stationären Rechner zur Verfügung haben. PDAs und Handys können über die Portalschnittstellen auf die verschiedenen Telearbeitsdienste zugreifen, wobei die Darstellung und auch die bereitgestellte Funktionalität an das spezielle Endgerät angepasst werden können.

Für bestimmte Benutzergruppen, wie z.B. Blinde und Sehgeschädigte, ist die Nutzung weiterer Schnittstellen wie VoiceXML zur Ausgabe von Informationen als auch zur Steuerung von Diensten und Anwendungen sehr wichtig. Auch für Monteure im Außendienst ist die Nutzung solcher Schnittstellen denkbar, wenn sie z.B. eine Information aus dem Hilfesystem des Herstellers abrufen wollen, obwohl sie keine Hand für die Bedienung eines Rechners frei haben. Viele Portalhersteller haben solche Schnittstellen bereits in ihrer Produktpalette aufgenommen, wie z.B. IBM mit *WebSphere Everyplace Mobile Portal* oder SAP *Netweaver* mit *Multi Channel Access*.

KAPITEL 6
VALIDIERUNG DER ENTWICKELTEN LÖSUNG

„Nichts ist so praktisch wie eine gute Theorie“

Kurt Lewin (1890-1947)

Die vorgeschlagene Lösung wurde sowohl durch eine prototypische Implementierung als auch durch verschiedene Fallstudien evaluiert, die im folgenden Kapitel beschrieben werden. Abschließend wird die entwickelte Lösung dahingehend bewertet, wie sie die in Kapitel 2.6 definierten Anforderungen erfüllt und welche Vorteile diese gegenüber vorhandenen Ansätzen bietet.

6.1 Prototypische Implementierung

Zur funktionalen Evaluation der entwickelten Lösung wurde eine prototypische Implementierung eines Telearbeitsportals und einzelner Telearbeitsdienste vorgenommen, um die praktische Umsetzbarkeit der vorgestellten Konzepte zu demonstrieren. Exemplarisch wurden einzelne Basisdienste implementiert oder vorhandene Web Services genutzt, um die Beschreibung der Schnittstellen und die Integration in das Portal zu zeigen. Die Realisierung der Basisdienste wurde bereits in Kapitel 5.2 ausführlich dargestellt.

Auf die Implementierung eines Autorenwerkzeuges zur Komposition der Telearbeitsdienste aus verschiedenen Basisdiensten wurde im Rahmen dieser Arbeit verzichtet, da auf dem Markt eine Reihe von Werkzeugen existieren, die die Komposition von komplexen Web Services auf Basis von BPEL bereits ermöglichen. Um den Entwicklungsprozess der Telearbeitsumgebung durchgehend unterstützen zu können, wäre die Integration eines graphischen Kompositionstools in das Framework zur Suche und Auswahl der Basisdienste natürlich wünschenswert, konnte im Rahmen dieser Arbeit aber nicht implementiert werden.

Um die flexible Dienstintegration in das Portal zu ermöglichen, wurden verschiedene Portlets implementiert, die die Suche nach geeigneten Web Services und die Erzeugung einer GUI für die aufgerufenen Web Services zur Aufgabe haben.

6.1.1 Portallösung

Im Rahmen dieser Arbeit wurde ein Prototyp des Telearbeitsportals umgesetzt. Dazu wurden verschiedene Portalframeworks sowohl aus dem kommerziellen als auch aus dem Open-Source-Bereich (OS) auf Ihre Eignung im Anwendungskontext untersucht und getestet. Grundvoraussetzung für die Nutzung von WSRP ist die Einhaltung des JSR168-Standards (siehe auch Kapitel 4.3.2). Tabelle 6-1 zeigt eine Aufstellung der Portalimplementierungen, die diesen Standard bisher umgesetzt haben und deshalb für die Nutzung in dieser Arbeit in Frage kommen.

Bisher wurden die Open-Source-Produkte *Apache JetSpeed*, *uPortal* und *Liferay* installiert und bezüglich ihrer Nutzbarkeit im Rahmen der vorgeschlagenen Lösung getestet. Das Universitätsprojekt uPortal scheint technologisch am weitesten gereift zu sein, hat jedoch nur ein sehr spärliches Angebot an vorgefertigten Portlets. Liferay hat besonders gut ausgereifte Portlets im Bereich der Kollaborationswerkzeuge wie Kalender, Seitengestaltung und Wiki. Jetspeed bietet vor allem konfigurierbare Ticker und Aggregatoren an. Ein

Austausch von JSR168-standardisierten Portlets ist entweder über entsprechende WAR-Dateien oder über WSRP lösbar. Jedoch unterstützt nur Liferay bisher die Integration von WSRP-Portlets, und außerdem gibt es bisher noch keine öffentlichen Verzeichnisse, aus denen man verfügbare Portlets auswählen könnte.

Anbieter	Produkt	Link
Apache (OS)	Jakarta JetSpeed 1.6	http://jakarta.apache.org/jetspeed/
BEA	Web Logic Portal 4.0	http://www.bea.com/products/weblogic/portal/
IBM	WebSphere Portal 2.1	http://www-4.ibm.com/software/webservers/portal/
iPlanet	iPlanet Portal Server 3.0	http://www.iplanet.com/products/iplanet_portal/
Oracle	Oracle 9i Portal	http://www.oracle.com/ip/deploy/ias/portal/
Liferay (OS)	Liferay Portal	http://www.liferay.com/
JA-SIGig (OS)	uPortal	http://www.uportal.org/

Tabelle 6-1: Übersicht über JSR168-konforme Portalimplementierungen

Die im Anschluss detaillierter beschriebenen Portlets zur flexiblen Integration von Web Services in Portale wurden mit JetSpeed 1.6 "Fusion" implementiert und gestestet. Abbildung 6-1 zeigt eine Portalseite mit einigen Beispiel-Portlets.

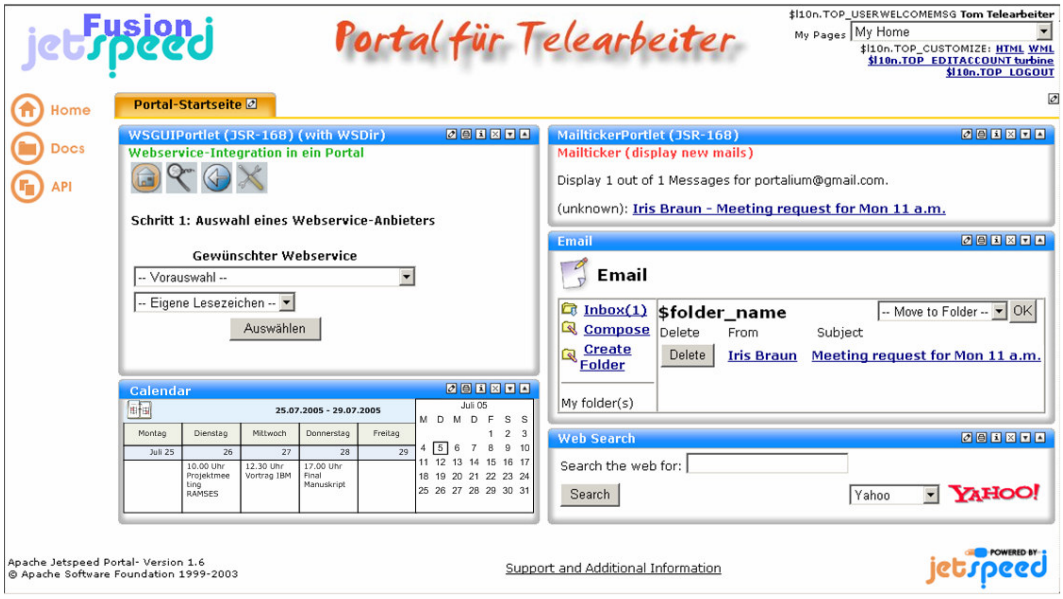

Abbildung 6-1: Portalseite in JetSpeed Fusion

Bei der Implementierung der Portlets wurde auf die Standardkonformität geachtet, um die Herstellerabhängigkeiten zu minimieren und eine Integration in andere Portalplattformen zu ermöglichen. Für das entwickelte `WSGUIPortlet` konnte die Integration in Liferay erfolgreich getestet werden.

Jedes Portlet kann dabei, sofern einmal installiert, mehrfach instanziiert werden, wobei jede Instanz einen eigenen inneren Zustand hat, der auch über mehrere Zugriffe hinweg beibehalten wird. Einige Aspekte davon wie Fenstergröße oder Modus (Ansicht, Konfiguration, Hilfe) werden dabei durch das Portal kontrolliert, alle anderen durch das Portlet selbst.

6.1.2 Flexible Dienstintegration

Im Rahmen der vorliegenden Arbeit wurden prototypisch einige Portlets im JetSpeed-Portal implementiert, um die Möglichkeiten zur Dienstintegration in ein Portal zu testen und zu veranschaulichen. Im Ergebnis entstand das Portlet `WSGUIPortlet`, welches die flexible Einbindung von Web Services in ein Portal ermöglicht. Es vereinigt die Funktionen des `WSUIPresentationPortlet` und die Suchfunktionalität aus dem `WSDirPortlet`, um einen kompletten Arbeitsablauf darstellen zu können. Mit dem `WSGUIPortlet` lassen sich Web Services aus einer internen Registry auswählen, in einem öffentlichen UDDI-Register suchen, in Favoriten-Listen zusammenstellen und direkt aufrufen. Abbildung 6-2 zeigt die verschiedenen Funktionen des Portlets in einem Use-Case-Diagramm.

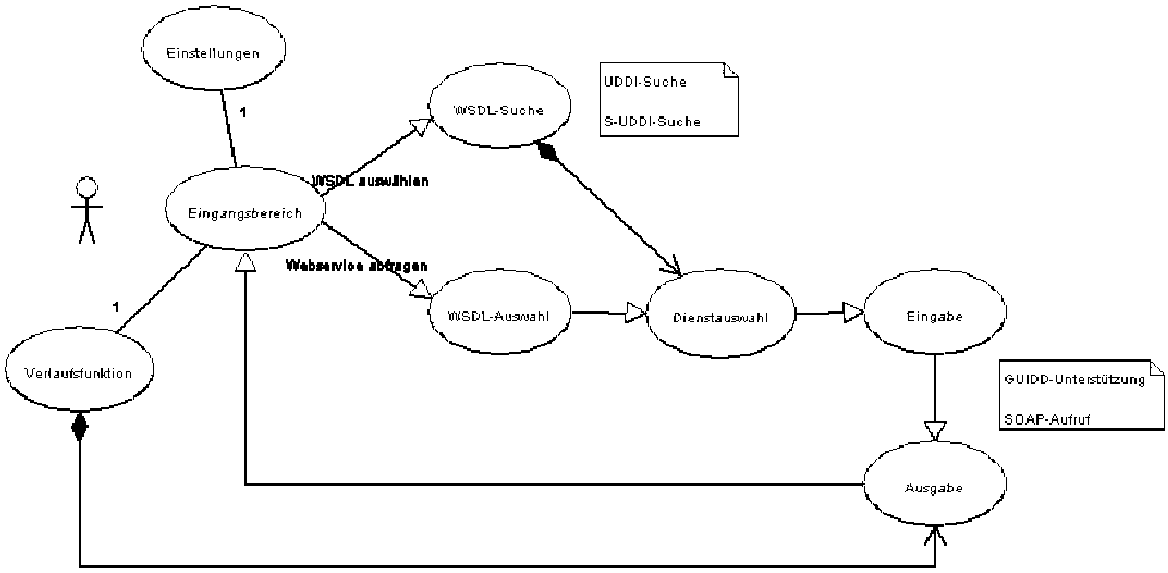

Abbildung 6-2: Use-Case-Diagramm des `WSGUIPortlet`

Das Portlet ist in Java implementiert und benötigt die Bibliotheken UDDI4J[54] und Apache SOAP[55]. Neben den SOAP-Aufrufen wurde auch das Einlesen von WSDL-Dateien direkt im Portlet implementiert. Dabei werden sowohl Dienste nach dem Web-Service-Standard basierend auf UDDI, WSDL und SOAP als auch REST-basierte[56] Dienste wie Open-Search[57] unterstützt. Um Open-Search-Dienste nutzen zu können, muss eine Open-Search-Description-Datei (OSD) vorhanden sein, die in etwa mit einer WSDL zu vergleichen ist, wobei das Ausgabeformat stets RSS ist. Zusätzlich wurde die Unterstützung des Formates GUIDD untersucht und implementiert, welches die Generierung von Benutzeroberflächen für Web Services zur Aufgabe hat (WSGUI-Konzept) und in Kapitel 4.2.5 bereits ausführlich beschrieben wurde.

---

[54] UDDI4J ist eine Java-Klassenbibliothek, die eine API zur Verfügung stellt, um mit einem UDDI-Register zu interagieren. (http://uddi4j.sourceforge.net/)

[55] Apache SOAP ist eine Implementierung des W3C-Standards, welche auf der IBM-Implementierung SOAP4J basiert. (http://ws.apache.org/soap/)

[56] REST (REpresentational State Transfer) ist eine Alternative zu SOAP und verwendet HTTP-Methoden, die auf URI-adressierbaren Ressourcen arbeiten. [OIO05]

[57] http://opensearch.a9.com/spec/opensearchdescription/1.0/

Zu dem Portlet gehören zwei Bibliotheken, die jeweils portletspezifische und generische Java-Klassen und Methoden enthalten. Darauf setzt die Portletklasse selbst auf. Die erste Bibliothek ist `wsdir` mit den Klassen `WSDirView`, `WSDirSearch` und `WSDirPreferences`. Die Klasse `WSDirView` ist für die HTML-Ausgabe zuständig. So wird das Formular zur Suche von der `doView()`-Funktion des Portlets aufgerufen, während andere Ausgaben (Liste der Lesezeichen, Suchergebnisse) intern oder extern individuell verwendet werden können. `WSDirSearch` implementiert eine lokale Suche über S-UDDI sowie eine entfernte Suche in öffentlichen UDDI-Registern. Die Klasse `WSDirPreferences` lädt und speichert die Lesezeichen je nach Parametern des Portletaufrufs.

Die zweite Bibliothek ist `wsgui`, welche intern auch `wsdir`-Klassen, z.B. für die Anzeige von Einträgen einer S-UDDI-Datei, verwendet. Sie hat vier Klassen: `WSGUIView` liefert HTML-Ausgaben zurück, also Debug-Ausgaben, HTML-Formulare und Auswahlmenüs für die WSDL-Navigation. `WSGUIParser` enthält Methoden zum Einlesen von WSDL-Dateien, OSD-Dateien und GUIDD-Dateien. Die Klasse `WSGUIQuery` kann SOAP-Anfragen bzw. REST-Anfragen an den Web Service stellen und die Antwort einlesen. Und schließlich ist `WSGUICore` die zentrale Klasse mit dem Session-Management der WSDL-OSD-Navigation sowie den Querverweisen auf die WSDL-Suche. Nach jedem Web-Service-Aufruf wird ein Eintrag in der Verlaufstabelle angelegt. Diese wird von `WSGUIHistory` verwaltet und soll perspektivisch Eingabedaten und Ausgabedaten mit abspeichern.

Das Portlet selbst heißt `WSGUIPortlet168.java` und hat alle nutzbaren Portletmethoden reimplementiert: `doView()` zum Anzeigen des Portlets, `doEdit()` zum Konfigurieren und `processAction()` zur Auswertung von HTTP-Parametern. Alle Anfragen werden nahezu unverändert direkt an `WSGUICore` bzw. `WSDir`-Klassen übermittelt. Ein UML-Klassendiagramm, welches die Architektur von `WSGUIPortlet` und die Zusammenwirkung der insgesamt neun Klassen demonstriert, ist im Anhang zu finden.

Gleich nach dem Laden präsentiert sich das WSGUIPortlet mit einem graphischen Menü sowie einer Startseite (siehe Abbildung 6-3). Im Menü lassen sich bequem die Portletfunktionen WSDL-Dialogführung (zu Beginn standardmäßig ausgewählt), WSDL-Suche, vormals aufgerufene Dialoge (Verlaufsfunktion) und Konfiguration aufrufen.

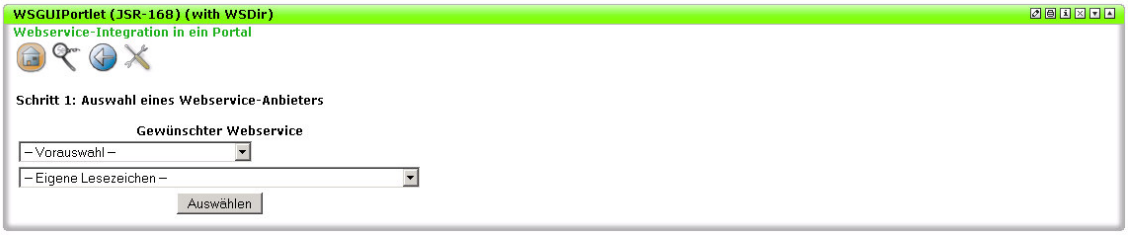

Abbildung 6-3: Startseite des WSGUIPortlet

6.1.2.1 Dienstsuche

Für die Suche nach geeigneten Telearbeitsdiensten, die der Telearbeiter in seine Arbeitsumgebung integrieren möchte, wurde prototypisch das Portlet `WSDirPortlet` implementiert, das sowohl in einer internen Registry als auch in einem öffentlichen UDDI-Register nach geeigneten Web Services sucht. Um die Entwickler und Anwender bei der Dienstsuche zu unterstützen, sollen die in Kapitel 5.3 vorgestellten semantischen Beschreibungen der Telearbeitsdienst-Ontologie in die Suchanfrage einbezogen werden können. Weiterhin hat der Telearbeiter die Möglichkeit, die Suche auf Dienste mit GUIDD-Beschreibung ein-

zuschränken (siehe Abbildung 6-4), also auf Web Services, für die bereits eine Beschreibung der Benutzeroberfläche angeboten wird. Diese Information wird ebenfalls in der internen Registry hinterlegt. Diese interne Registry, die für jeden Dienst Angaben zur WSDL-Datei und GUIDD-Datei sowie Name und Beschreibung enthalten kann, wird im Folgenden als S-UDDI ("*Simple UDDI*") bezeichnet. Außer WSDL-Dateien können auch OSD-Dateien darin vermerkt werden. Im Folgenden werden diese aber synonym als WSDL-Datei bezeichnet.

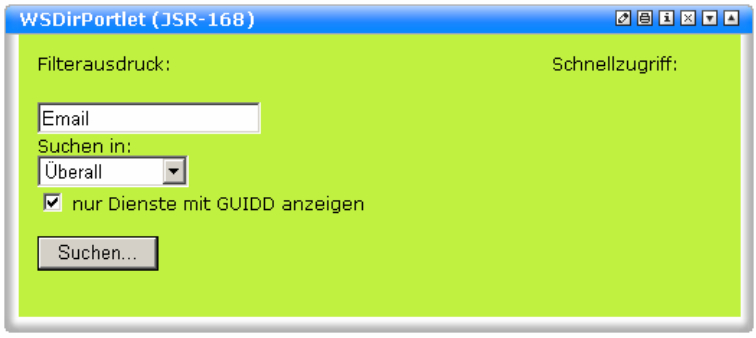

Abbildung 6-4: Suchanfrage in WSDirPortlet

Der interne Verzeichnisdienst S-UDDI ist eine simplifizierte Umsetzung einer UDDI. Er basiert auf einer XML-Datei, die für jeden Dienst je eine Zeile enthält, welche mindestens den Namen und die zugehörige WSDL-Datei referenziert.

```
<?xml version="1.0"?>
  <suddi>
    <wsdl file="dnslookupservice-rn.wsdl" name="DNS-Abfrage (über rn)" />
  </suddi>
```

Listing 6-1: Eintrag in S-UDDI

Darüber hinaus können Referenzen auf vorhandene GUIDD-Dateien angegeben werden.

```
<wsdl file="waiter.wsdl" name="Kellner" guidd="waiter-simple.guidd" />
```

Listing 6-2: GUIDD-Verweis in S-UDDI

Auch externe WSDL-Dateien werden unterstützt. Fehlt die Protokollangabe, wird automatisch eine lokale Datei im selben Verzeichnis wie das der S-UDDI-Datei angenommen.

```
<wsdl file="http://www.webservicex.net/sendsmsworld.asmx?wsdl" name="SendSMSWorld" />
```

Listing 6-3: Verweis auf externe WSDL

Zur besseren Klassifizierung können noch Schlüsselwörter bzw. eine Beschreibung angegeben werden, welche in der Suche dann mit berücksichtigt werden. Analog können auch die Eigenschaften, die über die Telearbeitsontologie beschrieben werden, eingefügt werden.

```
<wsdl file="email.wsdl" name="Email versenden" guidd="email.guidd"
 description="email,smtp,sendmail" isCategorizedBy="Kommunikationsdienst"/>
```

Listing 6-4: Beschreibung zusätzlicher Eigenschaften in S-UDDI

Parallel zur internen Registry wird ein öffentliches UDDI-Register durchsucht. Als UDDI-Verzeichnis wird momentan die IBM-Test-UDDI[58] verwendet, es wären aber auch andere

[58] http://www-3.ibm.com/services/uddi/testregistry/inquiryapi

möglich. Die Einstellung des verwendeten UDDI-Servers kann im `WSGUIPortlet` über die Konfiguration vorgenommen werden. Die anzuzeigenden Dienste werden mit Hilfe eines regulären Ausdrucks (standardmäßig `%send%`) lokalisiert. Die Abfrage erfolgt hierbei über die UDDI4J-Klassen, mit der Erweiterung, dass WSDL-Beschreibungen auch dann gesucht werden, wenn keine explizit angegeben ist, und zwar durch Anhängen von `'?wsdl'` an die so genannte `AccessPoint`-URL. Bei mehreren Anbietern scheint das der übliche Weg zu sein. Als Ergebnis erhält der Telearbeiter z.B. folgende in Abbildung 6-5 dargestellte Liste. Die Einträge aus der internen Registry sind mit S-UDDI gekennzeichnet.

Im Produktbetrieb sollte die Suche aber grundsätzlich auf die interne Registry beschränkt werden, damit der Telearbeiter nur Dienste zur Integration auswählen kann, die vorher von der IT-Abteilung des Unternehmens auf ihre Funktionsweise und Sicherheit getestet wurden. Ansonsten würde die Integration beliebiger Web Services aus unbekannten Quellen ein erhebliches Sicherheitsrisiko darstellen. Außerdem ermöglicht es den Entwicklern, zu den registrierten Web Services auch entsprechende GUI-Beschreibungen in der Registry zu hinterlegen.

Abbildung 6-5: Ergebnis einer Suchanfrage

Die gefundenen Dienste können dann direkt zur Ausführung ausgewählt werden. Der weitere Ablauf wird in den folgenden Kapiteln detaillierter beschrieben.

Zusätzlich wurde noch eine Lesezeichen-Funktion implementiert, die es dem Telearbeiter ermöglicht, bereits gefundene Dienste in einer Favoritenliste vorzumerken und später per Schellzugriff einfach aufzurufen (siehe Abbildung 6-5). Man kann aber auch anderweitig gefundene WSDL-Dateien direkt in die Favoritenliste integrieren, indem man die URL des gewünschten Web Service angibt.

6.1.2.2 Dienstauswahl

Neben der Suchfunktion wird auch die direkte Auswahl vorhandener Dienste ermöglicht, welche in der internen Registry verwaltet werden. Diese Vorauswahl ist standardmäßig als Startseite voreingestellt (siehe Abbildung 6-3), um die Wiederverwendung bereits bekannter Dienste zu erhöhen. Zusätzlich zur WSDL-Beschreibung wird auch die GUIDD-Beschreibung verlinkt. Somit kann der Nutzer sehen, welche Web Services bereits eine mittels GUIDD beschriebene Benutzeroberfläche anbieten. Hat der Anwender einen Dienst ausgewählt, werden die von ihm angebotenen Operationen angezeigt (siehe Abbildung 6-6). Nach erfolgter Auswahl wird das HTML-Formular zur Eingabe der Parameter erzeugt und angezeigt. Sowohl dort als auch schon vorher während der Navigation werden,

sofern eine GUIDD-Datei vorhanden ist, WSGUI-Informationen in die Seitengestaltung eingebracht. Nach Ausfüllen und Absenden des Formulars wird der Web Service aufgerufen und das zurück gelieferte Ergebnis wiederum in HTML ausgegeben.

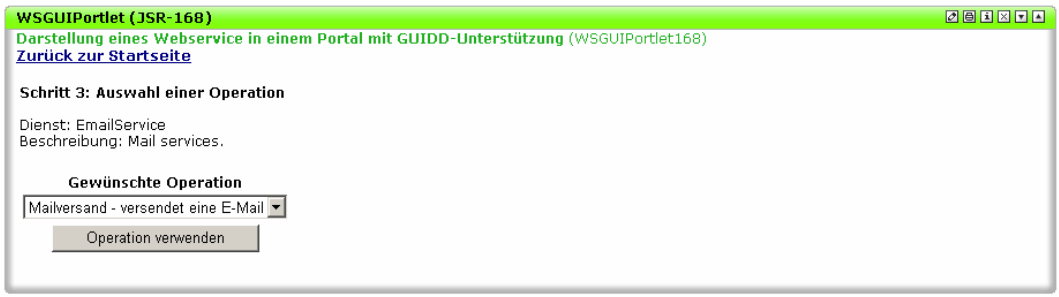

Abbildung 6-6: Auswahl der Operation eines Web Service

6.1.2.3 Dienstintegration

Grundsätzlich geht es darum, aus den vorhandenen WSDL-Schnittstellen eines Dienstes, die Ein- und Ausgabeparameter zu extrahieren und daraus eine Benutzerschnittstelle für den Web Service zu generieren, die in das Portal integriert werden kann. Bei der Implementierung des Prototypen wurden einige Entwicklungsstufen durchlaufen, die im Folgenden detaillierter beschrieben werden.

`WSUIPortlet`

Dieses Portlet wurde entwickelt, um eine Benutzerschnittstelle zu einem ausgewählten Web Service zu generieren. Es dient zur Darstellung der Funktionsweise und zum prototypischen Test der Integration in das Portal. Eine WSDL-Spezifikation wird von diesem Portlet herunter geladen, eingelesen und ausgewertet. Dabei wird ein HTML-Formular als Schnittstelle für den Web Service generiert und kann direkt im Portal genutzt werden. Das Portlet ist zum Test auf den Web Service `SendSMSWorld` festgelegt und nicht sehr flexibel bei der Anzeige. Es wurden aber bereits die Grundlagen für `WSUIExtendedPortlet` implementiert, wie die Verarbeitung des XML-Baumes durch den WSDL-Parser.

`WSUIExtendedPortlet`

Dies ist eine erweiterte Version von `WSUIPortlet`. Die WSDL-Spezifikation wird in zwei Durchläufen ausgewertet: im ersten Durchlauf werden alle komplexen Datentypdefinitionen zwischengespeichert und die Liste der möglichen Aufrufoperationen geladen, wobei der Einfachheit halber der erste mögliche Aufruf zur Anzeige ausgewählt wird. Im zweiten Durchlauf wird nun die mit dem ausgewählten Aufruf verknüpfte, möglicherweise mehrteilige Nachricht aufgesucht, und sämtliche benötigte Daten inklusive ihrer einfachen oder komplexen Typen dazu vermerkt. Abbildung 6-7 zeigt die Funktionsweise des Portlets, in dem die Debug-Ausgaben des WSDL-Parsers protokolliert werden. Anschließend erfolgt die Anzeige des Eingabeformulars, woraus die entsprechende SOAP-Nachricht zum Aufruf des Web Service generiert werden kann.

Ist das Formular ausgefüllt, kann es an den Web Service gesendet werden. Dabei wird nicht die URL des Web Service direkt kontaktiert, sondern die URL des Portlets selbst (bzw. der Seite, die es enthält). Dabei erkennt das Portlet, dass es nun kein Formular mehr anzeigen soll, sondern die bereits vorhandenen Eingaben an den Web Service weiterzureichen sind. Je nach Aufrufkonvention des Dienstes wird nun ein HTTP- bzw. SOAP-Aufruf getätigt. Der Web Service liest diese Daten aus und generiert dafür die Antwort oder einen Fehlercode. Das Ergebnis wird dem Nutzer wiederum als HTML-Formular angezeigt.

Abbildung 6-7: Durchläufe und Anzeige des WSUIExtendedPortlet

Das folgende Schema (Abbildung 6-8) erläutert noch einmal graphisch den Ablauf des Aufrufes eines Web Service im Portal.

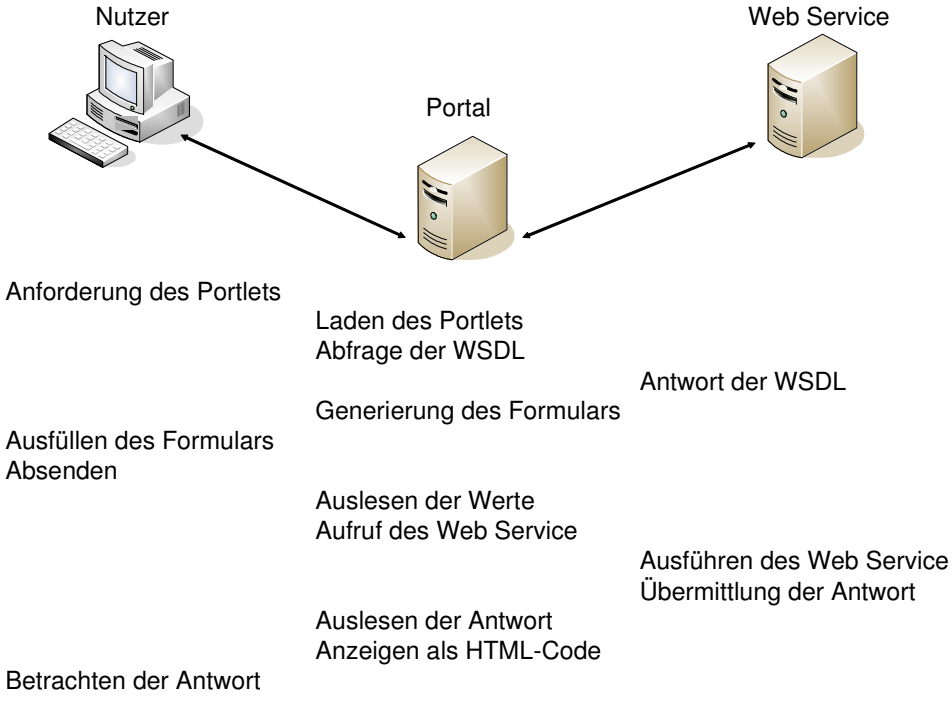

Abbildung 6-8: Schematischer Ablauf der Nutzung von WSUIExtendedPortlet

Es werden auch komplexe Datentypen unterstützt. Zu diesem Zweck hält das Portlet intern eine Liste aller komplexen Typen bereit, die zur Laufzeit (Anzeige oder Aufruf des Web Service) zu Wertelisten oder verschachtelten Werten expandiert werden. Des Weiteren

unterstützt es verschiedene Namespaces, rekursiv komplexe Typen und Variablen mit festen Wertzuweisungen.

`WSUIPresentationPortlet`

Dieses Portlet wurde auf der Grundlage des `WSUIExtendedPortlet` entwickelt, um die Ideen und Konzepte von GUIDD (siehe Kapitel 4.2.5) zu integrieren. Dabei wurde das an der University of Stanford entwickelte WSGUI-Konzept [KKM03] benutzt und weiterentwickelt. Es wurde aber eine grundlegend eigene Unterstützung für GUIDD implementiert, da sich die Implementierung aus Stanford als zu umfangreich auf der einen Seite und zu sehr auf Spezialfälle konzentriert auf der anderen Seite herausgestellt hat.

Abbildung 6-9 zeigt den prinzipiellen Unterschied zwischen der Generierung eines Formulars direkt aus der WSDL, wie von `WSUIExtendedPortlet` implementiert, und der Verwendung eines GUIDD, wie er von der WSGUI-Engine verwendet wird. Die WSGUI-Engine läuft dabei als Servlet in einer Containeranwendung (z.B. Tomcat) und bezieht sich primär auf das GUIDD-Dokument, in dem wiederum sowohl die WSDL als auch die Stylesheets, von denen es mehrere gleichzeitig geben kann, referenziert werden. Es wird ein Formular in purem XML erzeugt, ähnlich dem neuen W3C-Standard *XForms*, welches wiederum Referenzen auf die Stylesheets aufweist. Das Dokument wird an den Browser gesendet und durch die im Browser enthaltene CSS-Funktionalität (*Cascading Style Sheet*) entsprechend graphisch dargestellt.

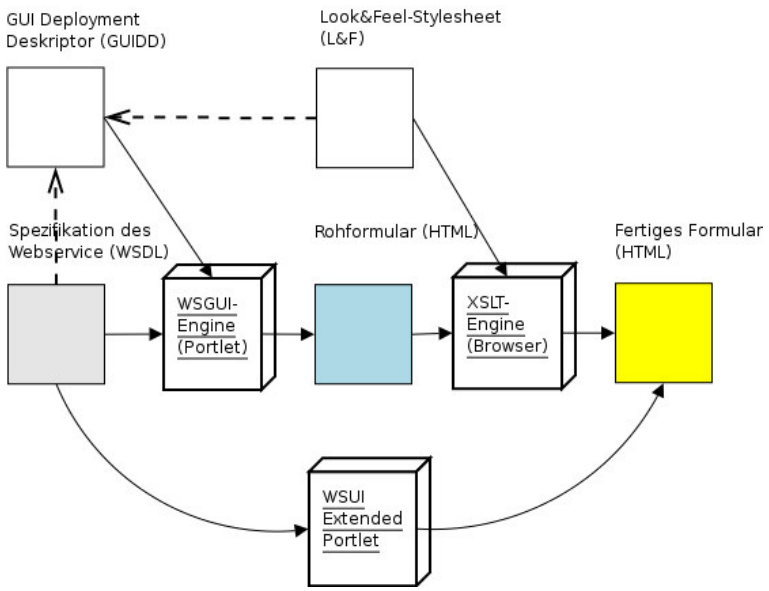

Abbildung 6-9: Nutzung eines GUI Deployment Deskriptor zur Formulargenerierung

Als Demonstrations-Web-Service wird unter anderem der in Kapitel 5.2.2.1 näher beschriebene E-Mail-Dienst genutzt. Dem Anwender wird nach der Auswahl des E-Mail-Services aus der Liste der verfügbaren Dienste und der Operation `SendEmail` ein Formular angezeigt, mit dem er eine E-Mail entwerfen und absenden kann (siehe Abbildung 6-10). Dabei werden anstatt der meist kryptischen Parameternamen die zugehörigen natürlich-sprachlichen Bezeichnungen aus dem GUIDD-Dokument angezeigt. Die WSGUI-Engine verknüpft dazu die internen WSDL-Variablennamen mit entsprechenden Elementen der GUIDD. Dadurch kann eine bessere Nutzbarkeit sowie Internationalisierung erreicht werden. Z.B. wird im angezeigten Formular für den Input `from` die deutsche Bezeichnung „Von“ und der Hilfetext „Ihre E-Mail-Adresse“ angegeben.

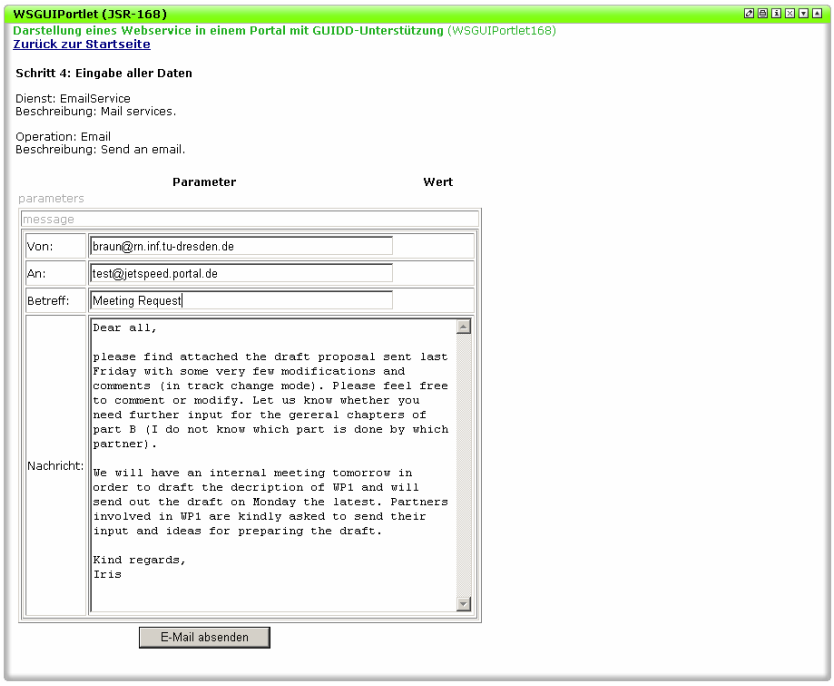

Abbildung 6-10: Beispiel für die Erzeugung einer GUI für den E-Mail-Service

Das zugehörige GUIDD-Dokument des E-Mail-Web-Service ist in Listing 6-5 dargestellt.

```
<?xml version="1.0" encoding="UTF-8" ?>
  <wsgui:deployment xmlns:wsgui="http://fxagents.stanford.edu/2002/10/wsgui"
   xmlns:targetns="urn:email" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
    <wsgui:wsdl href="email.wsdl" />
    <wsgui:formComponents>
      <wsgui:formComponent xpath="//xsd:element[@name='from']">
        <wsgui:input>
          <wsgui:label>Von</wsgui:label>
          <wsgui:help>Ihre E-Mail-Adresse.</wsgui:help>
        </wsgui:input>
      </wsgui:formComponent>
      <wsgui:formComponent xpath="//xsd:element[@name='to']">
        <wsgui:input>
          <wsgui:label>An</wsgui:label>
          <wsgui:help>Die E-Mail-Adresse, an die gesendet werden soll.</wsgui:help>
        </wsgui:input>
      </wsgui:formComponent>
      <wsgui:formComponent xpath="//xsd:element[@name='subject']">
        <wsgui:input>
          <wsgui:label>Betreff</wsgui:label>
          <wsgui:help>Betreffzeile der E-Mail.</wsgui:help>
          </wsgui:input>
      </wsgui:formComponent>
      <!--  basetype: GUIDD extension -->
      <wsgui:formComponent xpath="//xsd:element[@name='message']">
        <wsgui:input basetype="multiline">
          <wsgui:label>Nachricht</wsgui:label>
          <wsgui:help>Der Text der Nachricht.</wsgui:help>
        </wsgui:input>
      </wsgui:formComponent>
    </wsgui:formComponents>
  </wsgui:deployment>
```

Listing 6-5: GUIDD-Dokument des E-Mail-Service

Die Spezifikation von WSGUI wurde dahingehend erweitert, dass auch komplexere Formularelemente wie mehrzeilige Eingabefelder (siehe Abbildung 6-10) und Listenboxen dargestellt werden können. Zusätzlich kann man im GUIDD-Dokument auch die Zuordnung von Hilfetexten beschreiben und zwar mit dem `<wsgui:help>`-Element. In Abbildung 6-11 ist ein Beispiel für die Anzeige von Listenboxen und Hilfetexten zu sehen.

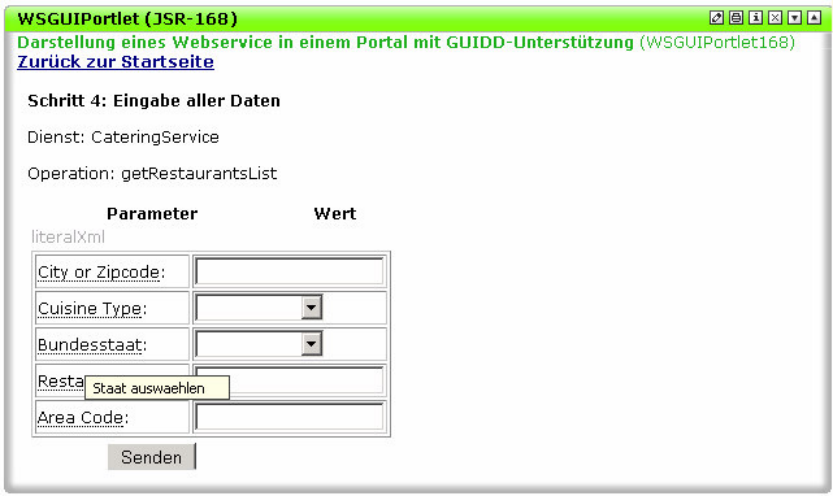

Abbildung 6-11: Darstellung von Listenboxen und Hilfetexten durch GUIDD

6.1.3 Bewertung der Implementierung

Mit der Implementierung des `WSGUIPortlet` konnte prototypisch gezeigt werden, wie die flexible Integration von Web Services in ein Portal erfolgen kann. Dabei wird sowohl die Suche neuer Services in einem öffentlichen Verzeichnis als auch die Wiederverwendung bereits registrierter Dienste ermöglicht. Die eigens entwickelte Registry S-UDDI kann auch mit semantischen Beschreibungen angereichert werden, womit die Suche effizienter gestaltet werden kann. Die Einbeziehung spezieller Ontologien wie die eigen entwickelte Telearbeitsdienst-Ontologie ist in einer späteren Ausbaustufe geplant.

Nach der Auswahl eines Dienstes und einer spezifischen Operation wird ein Eingabeformular für die Parameter des Web Service angezeigt. Um dieses Formular erzeugen zu können, wurde ein eigener WSDL-Parser entwickelt und implementiert. Dieser kann auch die GUIDD-Beschreibungen der Benutzeroberfläche des Web Service auslesen und bei der Darstellung des Eingabeformulars berücksichtigen. Die WSGUI-Spezifikation der Stanford University wurde im Rahmen der Arbeit um einige Funktionalitäten erweitert, um auch komplexe Datentypen darstellen zu können. Die komplette WSGUI-Spezifikation [Spi05] kann im Anhang nachgelesen werden.

Da GUIDD als Forschungsprojekt zwar auf einem Testsystem im Einsatz war, darüber hinaus aber nie verwendet wurde, sind einige noch fehlende Elemente in der Spezifikation aufgefallen. Zum einen ist es noch nicht direkt möglich, mehrsprachige Elemente zu verwenden, dazu muss momentan serverseitig anhand der Browsersprache eine GUIDD-Datei ausgewählt werden. Zum anderen wird die GUIDD-Datei in der Kopfzeile der WSDL-Datei verlinkt, wobei diese Beziehung eigentlich genau andersherum sein sollte, da GUIDD eine Erweiterung ist und man so problemlos neue Oberflächen zu existierenden Web Services hinzufügen könnte. Auch sind weitere Erweiterungen an WSGUI notwendig, so die Unterstützung von MIME-Typen für die Ausgabe, da dies bei den jetzigen Formularstandards noch nicht berücksichtigt ist.

Des Weiteren wäre zu klären, ob und inwieweit Menüführungen vorgegeben werden können. Oft unterstützt eine WSDL mehrere Operationen, die in einer bestimmten kausalen Reihenfolge aufgerufen werden müssen, wobei vorher empfangene Ergebnisse wieder mitgesendet werden. Für simple Aufgaben dieses Typs ist eventuell WSCL[59] geeignet, für

[59] WSCL (Web Service Conversation Language) ist eine XML-basierte Sprache zur Beschreibung erlaubter Nachrichtenfolgen, die Web Services austauschen dürfen.

komplexere Menüs wird man wohl um *FX-Agents*[60] oder andere derartige Konzepte nicht umhinkommen. Dieses Thema ist im Moment noch Forschungsgegenstand, weshalb bisher noch keine etablierten Methoden oder Protokolle dafür existieren.

Bei der Erstellung der automatisch generierten Formulare für die Eingabe und der ähnlich aufgebauten Ausgabeseiten gibt es mehrere Möglichkeiten bezüglich des Formats. Heutige Webseiten und Webbrowser kennen eigentlich nur die normalen HTML-Formulare, die auch als *Web Forms 1.0* bezeichnet werden. Bereits 2003 hat das W3C einen Standard namens *XForms* veröffentlicht [W3C03b], der weitaus komplexere Formulare ermöglichen soll. Dieser wurde aber von den Browserherstellern als zu schwierig eingestuft, vor allem in Hinblick auf die Autoren von Webseiten. Nun hat zwar Mozilla Anfang 2005 eine funktionierende XForms-Implementierung [MOZ05] veröffentlicht, dennoch haben sie sich zusammen mit Apple und Opera zur *Web Hypertext Application Technology Working Group* (WHAT) zusammengeschlossen, um aufbauend auf normalen HTML-Formularen den Standard *Web Forms 2.0* zu veröffentlichen [WHA05]. Dem XForms-Standard soll vielmehr eine Rolle für höherschichtige Geschäftsanwendungen und dergleichen zukommen. Andererseits gibt es mit *XForms Basic* nun auch eine simplere XForms-Variante. Die endgültige Verbreitung, die wohl über die Browserunterstützung definiert werden wird, ist im Moment noch nicht abzusehen.

Ein verwandtes Thema ist die generelle Beschreibung von Benutzeroberflächen durch XML-Dateien. Neben XUL[61] (Mozilla) und XML-Dialogen wie im *Qt-Toolkit*[62] wird auch Microsoft mit XAML[63] eine solche Technologie unterstützen. Eine Anpassung der entwickelten Lösung an XUL oder XAML wurde wegen der bisher unübersichtlichen Entwicklungslage bisher nicht vorgenommen, könnte aber als sinnvolle Erweiterung des Konzeptes im Anschluss an diese Arbeit untersucht werden.

Die Konsequenz aus den verschiedenen Entwicklungsmöglichkeiten ist die Modularisierung und Umwandlung des bisherigen `WSUIPresentationPortlet` in `WSGUIPortlet168`, das die eigentliche WSDL-Parsefunktion enthält, und einer JAR-Datei, die verschiedene Ausgabeformen unterstützt.

Mit GUIDD können bisher nur Ein- und Ausgabeformulare erzeugt werden, aber keine komplexeren GUIs (z.B. Office-Anwendungen) dargestellt werden. Hier wäre folgender Kompromiss möglich. Für die Verwaltung der Dokumente werden Kollaborationsdienste im Portal genutzt (z.B. der in Kapitel 5.2.3.1 beschriebene Dokumentenmanagementdienst), die eigentliche Bearbeitung der Dokumente findet aber temporär auf dem Client mit vorhandenen Anwendungen statt. Ein sehr ähnliches Konzept verfolgt Microsoft mit den so genannten Sharepoint Services. Dort werden in einem Portal alle notwendigen Verwaltungsdienste für Dokumente und Projekte zur Verfügung gestellt. Soll das Dokument bearbeitet werden, werden auf dem Client-Rechner installierte MS-Office-Werkzeuge benutzt. Der Nachteil dieses Kompromisses ist das Aufbrechen des Thin-Client-Ansatzes und damit die Verringerung der Wartbarkeit und Sicherheit. Die Sicherheit der Dokumente auf dem Telearbeitsrechner muss dann separat gewährleistet werden.

---

[60] http://fxagents.stanford.edu/

[61] XUL (XML User Interface Language) ist eine XML-basierte Beschreibungssprache für graphische Benutzeroberflächen (GUI). Sie wurde ursprünglich für das Mozilla-Projekt entwickelt.

[62] Das Qt-Toolkit ist eine Klassenbibliothek und Entwicklungsumgebung für die plattformübergreifende Programmierung graphischer Benutzeroberflächen (GUI) unter C++.

[63] XAML (eXtensible Application Markup Language) ist ein Konzept von Microsoft, das ähnlich XUL die generische Beschreibung von GUIs zur Aufgabe hat.

Eine weitere Lösung zur Integration komplexerer Oberflächen wäre die Integration externer Portlets in das Portal mit Hilfe von WSRP (siehe Kapitel 4.3.4). Somit wäre die Integration vorhandener Portlets anderer Anbieter möglich, was die Kosten für die Entwicklung eigener Portlets senken kann. Weitere Vorteile ergeben sich z.B. im Szenario der On-Site-Telearbeit. Ein Telearbeiter könnte sich so ganz einfach verschiedene Portlets aus dem Mitarbeiterportal des Kunden (z.B. Kalender, News, etc.) integrieren und so besser in die Projektkommunikation eingebunden werden. Der Vorteil für den Telearbeiter liegt darin, dass er alle benötigten Portlets in einer ihm vertrauten Oberfläche zusammenfassen kann und nicht auf verschiedene Portale zugreifen muss.

6.2 Evaluierung in verschiedenen Anwendungsszenarien

Anhand verschiedener Fallstudien soll der Mehrwert, der durch die Nutzung der Telearbeitsumgebung entsteht, analysiert und evaluiert werden. Eine wichtige Anforderung an die vorgeschlagene Lösung ist die Universalität. Die Arbeitsumgebung soll in der Hinsicht universell sein, dass sie für alle Telearbeitsformen und auch im „normalen“ Büroalltag gleichermaßen nutzbar ist. Der Aufbau der Arbeitsumgebung soll so flexibel sein, dass man sie an alle erdenklichen Einsatzszenarien anpassen kann. Deshalb werden in den folgenden Abschnitten die Nutzeffekte der vorgeschlagenen Lösung exemplarisch für drei verschiedene Telearbeitsformen näher untersucht.

6.2.1 Alternierende Telearbeit

Als erstes Anwendungsszenario wurde die alternierende Telearbeit ausgewählt, da sie mit Abstand die am weitesten verbreitete Telearbeitsform ist. Bei alternierender Telearbeit, auch Telependeln genannt, findet ein Wechsel zwischen dem Arbeitsplatz im Büro und zu Hause statt. Dabei nutzt der Telearbeiter zu Hause meist auch einen fest installierten PC-Arbeitsplatz. Die alternierenden Telearbeiter haben oft ein festgelegtes Aufgabengebiet, das sie im Wechsel im Büro oder zu Hause bearbeiten. Die benötigten Anwendungen und Daten auf dem Telearbeitsrechner entsprechen denen am Büroarbeitsplatz. Das Hauptproblem bei alternierender Telearbeit ist die Synchronisation der Daten auf dem Büro- und dem Heimarbeitsrechner. Diese Synchronisation kann beim Einsatz der vorgeschlagenen Lösung entfallen, da immer auf der gleichen Datenbasis gearbeitet wird. Abbildung 6-12 zeigt das Anwendungsszenario Alternierende Telearbeit - links ohne Einsatz der entwickelten Lösung und rechts mit Nutzung des Telearbeitsportals.

Da die Arbeitsabläufe alternierender Telearbeiter weitestgehend bekannt und starr sind, kann der Administrator die Telearbeitsumgebung entsprechend vorkonfigurieren. Die benötigten Telearbeitsdienste, die die Arbeitsabläufe des Telearbeiters abbilden, können von der IT-Abteilung des Unternehmens aus Basisdiensten komponiert und in einer internen Registry verwaltet werden. Der Administrator konfiguriert das Portal für den Telearbeiter entsprechend vor, indem er die benötigten Telearbeitsdienste und weitere Basisdienste bereits integriert. Diese Vorkonfiguration ermöglicht außerdem, dass verschiedene Telearbeiter den gleichen Satz an vorgegebenen Telearbeitsdiensten nutzen können. Dadurch wird die Wartung deutlich vereinfacht, denn der Administrator kann benötigte Dienste zentral aktualisieren und hinzufügen. Verträge mit Fremdanbietern von Basisdiensten können zentral ausgehandelt und abgeschlossen werden. Dabei wird die Zuverlässigkeit der Dienste durch den Provider vertraglich zugesichert, was die Verfügbarkeit des Gesamtsystems erhöht.

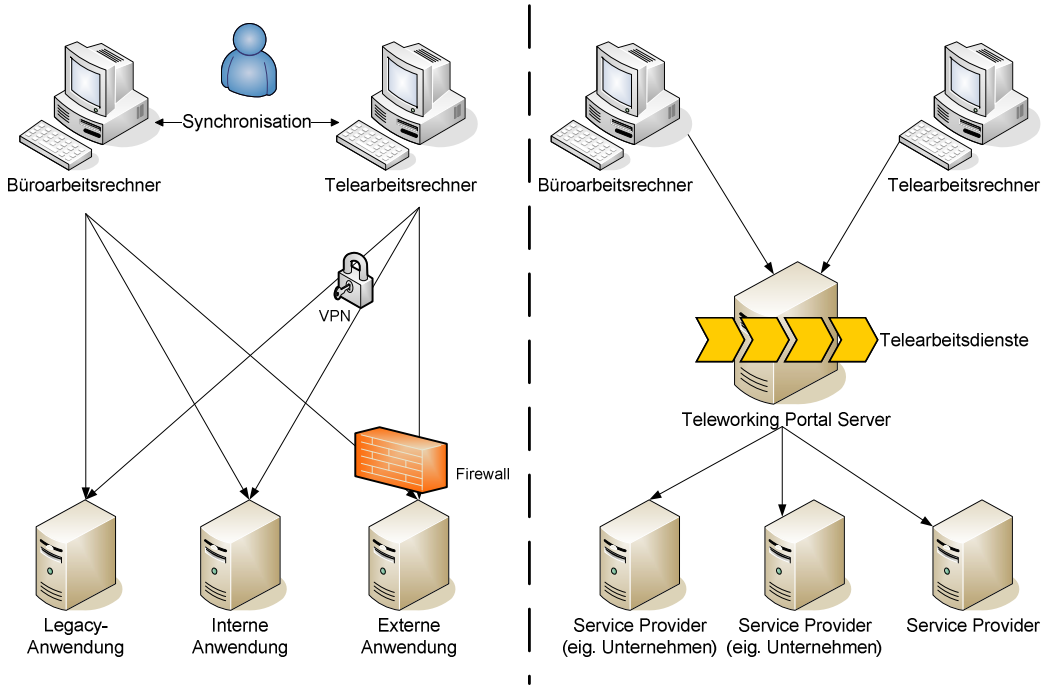

Abbildung 6-12: Anwendungsszenario Alternierende Telearbeit

Der Telearbeiter hat aber auch die Möglichkeit, selbst weitere Dienste zu integrieren. Dabei hat er zwei Alternativen. Er kann Dienste integrieren, die in der internen Registry seines Unternehmens verwaltet werden. Diese Services wurden durch die IT-Abteilung bezüglich Funktionsweise und Sicherheit getestet. Alternativ dazu wäre auch eine Integration externer Web Services möglich, die durch eine UDDI-Suchanfrage gefunden werden können. Diese Option sollte aber nur sehr erfahrenen Nutzern zur Verfügung gestellt werden, da sie ein großes Sicherheitsrisiko in sich birgt. Durch ein entsprechendes Rollenmanagement kann die Flexibilität für verschiedene Nutzergruppen eingeschränkt werden. Für unerfahrene Nutzer sollte auf jeden Fall eine Beschränkung auf interne Dienste und Anwendungen vorgenommen werden. Theoretisch wäre es auch möglich, die komplette Integration neuer Dienste zu verbieten, dies würde aber die Flexibilität der Lösung und die Möglichkeiten der Individualisierung deutlich einschränken.

Durch den personalisierten Zugang zum Portal wird ein hinreichender Datenschutz gewährleistet, da nur angemeldete Nutzer auf die Telearbeitsumgebung und die dahinter liegenden Anwendungen und Daten zugreifen können. Durch das Thin-Client-Prinzip werden keine Firmendaten auf den Heimrechner geladen, sondern zentral im Unternehmen oder auf Seite des Service Providers verarbeitet. Damit wird die Gefahr, dass vertrauliche Daten von Familienmitgliedern oder Dritten gelesen werden können, minimiert. Außerdem ist eine deutliche Trennung von privater und dienstlicher Nutzung des Rechners möglich. Dies kann große Kostenvorteile bringen, da die Telearbeiter meist eigene private Rechner besitzen und diese dann problemlos für Telearbeit nutzen können. Meist müssen durch den Arbeitgeber dann anstatt der Beschaffung eines Neurechners nur geringe Abnutzungsgebühren erstattet werden. Auch die Installation und Wartung von Anwendungssoftware entfällt, da am Telearbeitsplatz nur ein Internetzugang und ein Browser notwendig sind. Dies reduziert den Wartungsaufwand erheblich.

Weitere Nutzeffekte ergeben sich bei einem Einsatz des Systems auch am Büroarbeitsplatz. Erstens muss sich der Telearbeiter dann nicht mehr zwischen zwei unterschiedlichen Systemen umstellen und zweitens ermöglicht dies den Einsatz von variablem Desk-Sharing, also die Nutzung von Büroarbeitsplätzen durch mehrere Mitarbeiter, da keine spezielle Anwendungssoftware installiert werden muss und die Daten auf dem Server gehalten

werden. Somit verringert sich auch der Einarbeitungsaufwand, wenn Mitarbeiter in Telearbeit wechseln möchten, und eine kostengünstige zentrale Schulung für alle Mitarbeiter wird möglich.

6.2.2 On-Site-Telearbeit

Als nächster Anwendungsfall wurde On-Site-Telearbeit betrachtet. In diesem Szenario arbeitet der Telearbeiter zeitweilig direkt bei einem Kunden oder Partner. Dies ist z.B. denkbar, wenn ein Softwareentwickler für einige Monate vor Ort beim Kunden arbeitet oder mehrere Partner für die Laufzeit eines Projektes ein gemeinsames Büro nutzen. Das Besondere dieses Szenarios ist die Nutzung eines Arbeitsplatzrechners im Unternehmen des Kunden oder Partners. Der Telearbeiter soll aber trotzdem die Möglichkeit haben, auf alle benötigten Anwendungen und Daten seines eigenen Unternehmens zugreifen zu können. Wie dies mit der vorgeschlagenen Lösung erreicht werden kann, ist in Abbildung 6-13 auf der rechten Seite dargestellt. Auf dem Kundenrechner müssen nur ein Internetzugang und ein Browser vorhanden sein, damit der Telearbeiter auf das Telearbeitsportal seines Unternehmens zugreifen kann. Hier stehen ihm die bekannten Integrationsmöglichkeiten zur Verfügung und vorkonfigurierte Telearbeitsdienste sind bereits vorhanden.

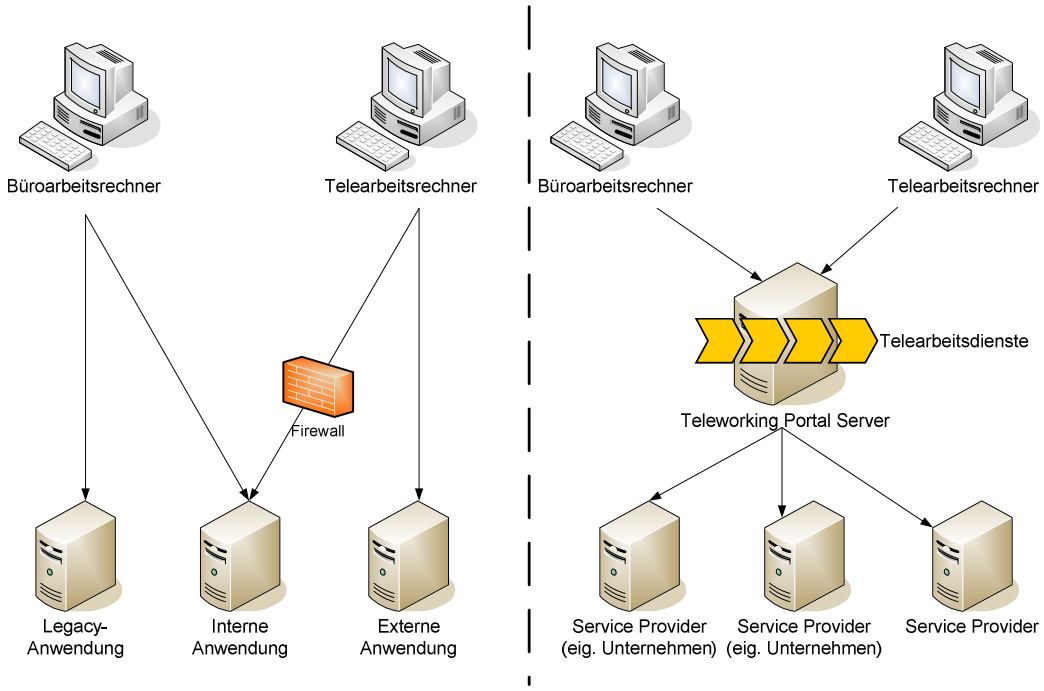

Abbildung 6-13: Anwendungszenario On-Site-Telearbeit

Weiterhin kann der Telearbeiter zusätzliche Dienste integrieren, um eine bessere Einbindung in die Kommunikation beim Kunden oder eine einfache Einbeziehung von Daten und Informationen aus den Systemen und Anwendungen des Kunden zu ermöglichen. Er könnte z.B. den Gruppenkalender des Projektteams in seine Arbeitsumgebung integrieren, um in die Meeting-Planung einbezogen zu werden. Ist der Telearbeiter nicht so erfahren, dass er selbst Web Services des Kunden einbinden kann, wäre auch eine Zusammenarbeit der IT-Abteilungen beider Unternehmen denkbar, um die Geschäftsprozesse und Arbeitsabläufe auf entsprechende Telearbeitsdienste abzubilden und sie dem Telearbeiter zur Integration in seine Arbeitsumgebung anzubieten. Grundvoraussetzung dafür ist, dass der Kunde oder Partner Web-Service-Schnittstellen zu seinen Anwendungen bereitstellt. Die Flexibilität der Einbindung von Kunden-Diensten ist ein großer Vorteil der Lösung im Anwendungsszenario On-Site-Telearbeit.

Auch in diesem Szenario gibt es einige Sicherheitsrisiken durch die Nutzung eines externen Rechners. Diese können aber durch den personalisierten Zugang zum Portal ausgeschlossen werden. Da keine internen Firmendaten auf dem Fremdrechner abgelegt werden, ist ausreichender Datenschutz gewährleistet. Es besteht somit keine Gefahr, dass Daten ausspioniert werden, wenn der gleiche Computer auch durch Angestellte des Kunden oder Partners genutzt wird.

6.2.3 Mobile Telearbeit

Mobile Telearbeit stellt ein besonders anspruchsvolles Szenario dar, was die Flexibilität und Adaptierbarkeit der Telearbeitsumgebung betrifft. Mobile Telearbeiter nutzen die verschiedensten Endgeräte für ihre ortsunabhängige Arbeit. Bisher mussten sie sich selbst um die Synchronisation der unterschiedlichen Geräte und Anwendungen kümmern. Mit bestimmten Endgeräten konnten sie auch nur auf spezielle Anwendungen zugreifen; beim Wechsel des Endgerätes ging der Kontext der vorangegangenen Arbeiten verloren oder musste explizit übertragen werden. Dieses Szenario wird in Abbildung 6-14 auf der linken Seite veranschaulicht.

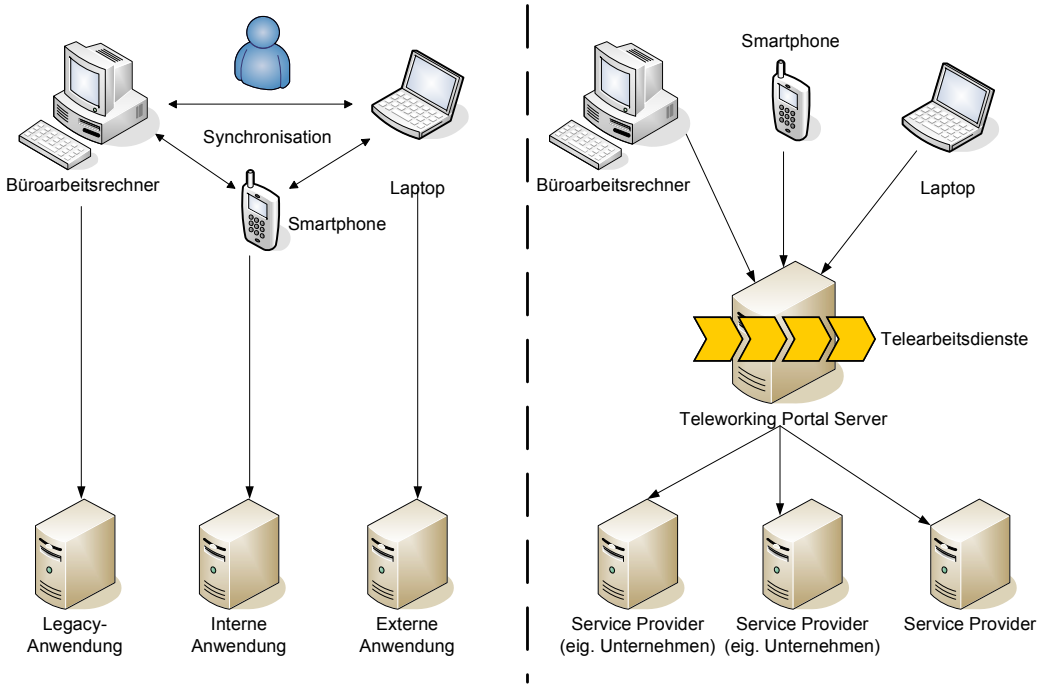

Abbildung 6-14: Anwendungsszenario Mobile Telearbeit

Für die Telearbeitsumgebung bedeutet das, dass sie sich flexibel an die verschiedenen Endgeräte und Modalitäten anpassen muss. Da die Telearbeiter oft bei Kunden und Partnern vor Ort sind, wäre es sinnvoll von denen angebotene Dienste in die Telearbeitsumgebung integrieren zu können, um z.B. einen schnellen Datenaustausch oder -abgleich zu ermöglichen. Dazu sollte der Portal-Admin die Telearbeitsumgebung des mobilen Telearbeiters mit einem Satz von vorkonfigurierten Telearbeitsdiensten bereitstellen, zu denen er weitere zusätzliche Dienste flexibel hinzufügen kann. Eventuell wird es im Vorfeld notwendig, dass die IT-Abteilungen des eigenen Unternehmens und des Kunden zusammenarbeiten und die gemeinsamen Geschäftsprozesse und Arbeitsabläufe auf komplexe Web Services abbilden. Diese neu entwickelten Telearbeitsdienste werden dem Telearbeiter dann über die interne Registry zur Einbindung in die Arbeitsumgebung zur Verfügung gestellt.

Die Adaption der Arbeitsumgebung an verschiedene Endgeräte wird durch das Portal übernommen, welches verschiedene Zugänge und Benutzeroberflächen für unterschiedliche

Endgeräte zur Verfügung stellt. So werden sowohl WML-Ein- und Ausgaben als auch Voice-XML-Schnittstellen von den meisten Portalherstellern unterstützt. Sollten diese Adaptionsmöglichkeiten nicht ausreichen, müssen die Telearbeitsdienste adaptiert oder entsprechend adaptierbare Dienste neu entwickelt werden.

Wenn mobile Endgeräte genutzt werden, muss natürlich sichergestellt werden, dass auch bei Verbindungsabbruch der Kontext der Arbeiten erhalten bleibt. Dies ist schon bei der Entwicklung der Telearbeitsdienste zu berücksichtigen und entsprechende Mechanismen sind in die Prozessabläufe einzubauen. Nutzbare Protokolle wären z.B. WS-Transaction[64] oder WS-Reliable-Messaging[65].

Vorteil des Einsatzes der vorgeschlagenen Lösung im Anwendungsszenario der mobilen Telearbeit ist die flexible Anpassung der Arbeitsumgebung an verschiedene Endgeräte und Modalitäten und die einfache Konfigurierbarkeit der genutzten Dienste.

6.3 Bewertung der vorgeschlagenen Lösung

In den folgenden Abschnitten wird ein Vergleich der vorgeschlagenen Lösung mit den in Kapitel 2.6 definierten Anforderungen an die Umsetzung einer universellen Telearbeitsumgebung durchgeführt und eine abschließende Bewertung der Lösung vorgenommen.

6.3.1 Abgleich mit den Anforderungen

Um eine Validierung der vorgeschlagenen Lösung durchführen zu können, wird zuerst ein Abgleich mit den in Kapitel 2.6 aufgestellten Anforderungen vorgenommen.

Universelle Einsetzbarkeit

In den vorigen Abschnitten konnte gezeigt werden, dass die vorgeschlagene Lösung für die verschiedenen Telearbeitsformen und Einsatzszenarien gleichermaßen geeignet ist und damit der Forderung nach einer universellen Einsetzbarkeit entspricht.

Modularer Aufbau

Mit der Umsetzung der serviceorientierten Architektur wird die Anforderung nach Modularität vollständig erfüllt, da die Arbeitsumgebung aus in sich gekapselten und unabhängigen Diensten zusammengesetzt wird. Dabei ist es völlig egal, in welcher Programmiersprache und auf welcher Plattform die Komponenten entwickelt wurden, sie können flexibel in die Arbeitsumgebung integriert werden.

Webbasierter Zugang

Der Zugriff auf die Arbeitsumgebung erfolgt durch einen Browser, also webbasiert. Die Lösung setzt damit den gängigen Thin-Client-Ansatz um, welcher eine Reihe von Vorteilen bietet. Auf dem Telearbeitsrechner entfällt die Installation von Programmkomponenten. Damit sind weder Umkonfigurationen des Rechners noch regelmäßige Updates der Client-Software notwendig. Die Telearbeitsrechner können insgesamt kostengünstiger ausgestattet werden, da teure große Festplatten zur Speicherung der Anwendungsprogramme und

[64] WS-Transaction - von IBM, Microsoft und BEA entwickelte Spezifikation, die auf das Coordination Framework von WS-Coordination aufbaut und die Koordination von Geschäftsprozessen und Transaktionen unterstützt. [IBM04]

[65] WS-Reliable-Messaging – von OASIS seit 2004 standardisierte Spezifikation, die den sicheren und zuverlässigen Nachrichtenaustausch zwischen Web Services garantiert. [OAS04b]

-daten entfallen und niedrige Prozessor-Taktraten ausreichen, da keine Anwendungslogik ausgeführt werden muss.

Prozessorientierung

Die Abbildung komplexer Geschäftsprozesse und Arbeitsabläufe erfolgt prozessorientiert. Mit Hilfe der Web-Service-Kompositionssprache WS-BPEL, die auf dem Orchestrationsansatz basiert, werden die Workflows auf abstrakte Prozesse abgebildet, die eine Reihe von Basisdiensten aufrufen. Bei der Komposition der BPEL-Prozesse wird der Entwickler durch ein Framework unterstützt, dass die Suche und Auswahl von Basisdiensten durch semantische Beschreibungen und eine Kategorisierung vereinfacht und effektiviert.

Kommunikations- und Kooperationsförderlichkeit

Die Telearbeitsumgebung bietet sowohl Dienste zur Kommunikation als auch zur Kooperation an. So wurde z.B. exemplarisch ein Dienst zum Versenden von E-Mails oder ein Kalenderdienst implementiert. Entsprechende Basisdienste werden den Kategorien der Kommunikations-, Koordinations- und Kollaborationsdienste zugeordnet und können in komplexere Geschäftsprozesse und Arbeitsabläufe integriert werden. Durch die prozessorientierte Kommunikation und Kooperation kann die Zusammenarbeit in Teams und Projektgruppen verbessert werden.

Standardisierung des Datenaustauschformats

Da Web Services komplett auf XML basieren, wird dieses universelle Datenaustauschformat in der entwickelten Lösung durchgängig benutzt. Dies ermöglicht den Datenaustausch zwischen völlig heterogenen Anwendungen unabhängig davon, in welcher Programmiersprache und auf welcher Betriebssystemplattform sie entwickelt wurden.

Individualisierbarkeit und Personalisierbarkeit

Die entwickelte Lösung zeichnet sich durch einen hohen Grad an Flexibilität und Anpassbarkeit aus. Es wurde ein Framework geschaffen, mit dem der Telearbeiter schnell und einfach verschiedenste Dienste und Anwendungen entsprechend seinen Anforderungen und Bedürfnissen integrieren kann. Durch die Portalframeworks werden eine Reihe von Personalisierungsfunktionen angeboten, die von der Anpassung des Look-and-Feel (Farben, Anordnung etc.) bis hin zur Ein- und Ausblendung von bestimmten Portlets und Anwendungen reichen.

Adaptierbarkeit

Mit dem Einsatz eines Portals als Benutzerschnittstelle werden die Möglichkeiten dieser Technologie genutzt, um eine Anpassung an verschiedene Endgeräte und Modalitäten zu erreichen. Heutige Portallösungen bieten Schnittstellen wie WML und Voice-XML, um die Adaption der Inhalte an mobile Endgeräte zu ermöglichen. Weiterhin wäre es denkbar, dass die Web Services selbst an das entsprechende Endgerät adaptiert werden, indem ein Eingabeparameter beim Aufruf des Dienstes die Kontextinformation zum verwendeten Endgerät enthält und der Dienste seine Funktionalität und Ergebnisse entsprechend anpasst. Ein E-Mail-Dienst könnte bei Aufruf über ein Handy z.B. nur die Header der E-Mails übertragen, um das Datenvolumen und die Anzeigegröße zu minimieren. Wird der gleiche Dienst von einem PC aufgerufen, werden die kompletten E-Mails und eventuelle Attachments herunter geladen.

Offenheit der Systemimplementierung

Mit der durchgehenden Nutzung standardisierter Protokolle und des XML-Standards zur Beschreibung der Schnittstellen sind Web-Service-basierte Lösungen grundsätzlich offene Systeme. Somit zeichnet sich auch die entwickelte Lösung auf Basis von Web Services durch die Offenheit der Implementierung aus.

Plattformunabhängigkeit

Ebenfalls auf Grund der durchgängigen Verwendung von XML ist die entwickelte Anwendung plattformunabhängig und kann damit auf alle Betriebssystemplattformen portiert werden. Die Client-Anwendung ist ebenfalls plattformunabhängig, da es sich dabei um einen auf allen Systemen vorhandenen Webbrowser handelt.

Verteilbarkeit des Systems

Serviceorientierte Architekturen setzen in Ihrem Ansatz grundlegend die Verteilung der einzelnen Dienste auf verschiedene physikalische Knoten voraus. Durch die lose Kopplung der Web Services können diese problemlos durch andere gleichwertige Dienste ersetzt werden.

Transparenz

Für den Telearbeiter bleibt es weitgehend transparent, wie die Telearbeitsdienste zusammengesetzt sind und von wem die Basisdienste erbracht werden. Komplexe Dienste sind über eine einzige Schnittstelle erreichbar, womit die dahinter liegende Komplexität und die verwendeten Komponenten dem Nutzer verborgen bleiben.

Skalier- und Erweiterbarkeit

Durch die Verteilung der Architektur auf mehrere Ebenen steigt die Skalierbarkeit des Systems und es lassen sich deutlich mehr Clients unterstützen als bei herkömmlichen 2- oder 3-Tier-Architekturen. Das System kann jederzeit flexibel um weitere Telearbeitsdienste erweitert werden.

Wiederverwendbarkeit

Durch die Komposition einfacher Basisdienste zu komplexen Telearbeitsdiensten kann die Wiederverwendbarkeit deutlich erhöht werden. Die einzelnen Basisdienste kapseln eine einfache, in sich geschlossene Funktionalität und können so in anderen Geschäftsprozessen und Arbeitsabläufen wiederverwendet werden. Das entwickelte System unterstützt die Wiederverwendung dadurch, dass bereits genutzte Basisdienste in einer internen Registry verwaltet werden und dem Nutzer zuerst für die Integration vorgeschlagen werden. Kann kein passender Dienst gefunden werden, wird nach neuen Web Services in weltweiten UDDI-Registern gesucht.

Verfügbarkeit

Durch die bereits erwähnte Verteilbarkeit kann die Verfügbarkeit des Gesamtsystems deutlich erhöht werden, da fehlerhafte oder ausgefallene Dienste leicht durch gleichwertige Dienste ersetzt werden können. Durch die Möglichkeit, konkrete Web Services erst zur Laufzeit zu binden, kann die Auswahl der Dienste auch aktuelle Quality-of-Service-Eigenschaften wie die Verfügbarkeit einbeziehen.

Zuverlässigkeit

Die Zuverlässigkeit des Gesamtsystems hängt natürlich von der Zuverlässigkeit der einzelnen Dienste ab. Diese kann man aber nur bedingt beeinflussen. Durch die flexible Integration kann aber ein Austausch von unzuverlässigen Diensten schnell und transparent durchgeführt werden. Außerdem können schon bei der Auswahl von Diensten Quality-of-Service-Eigenschaften einbezogen werden und mit dem Provider so genannte *Service Level Agreements* vereinbart werden.

Benutzbarkeit

Die Benutzbarkeit der Telearbeitsumgebung wird dadurch erhöht, dass die Telearbeiter die Umgebung an ihre individuellen Bedürfnisse und Vorlieben anpassen können. Bei der Gestaltung der Benutzeroberfläche des Portals wurden die gängigen Entwurfsmuster und User-Interface-Metaphern zugrunde gelegt. Auch der Zugriff über einen Webbrowser sollte allen Benutzern bekannt sein, was die Einarbeitungszeit deutlich reduziert.

Wartbarkeit

Die Nutzer benötigen nur einen Browser, um auf ihre Arbeitsumgebung zugreifen zu können. Dadurch entfallen Installation und Wartung von Anwendungsprogrammen auf dem Telearbeitsrechner. Durch die zentrale Speicherung der Nutzerprofile und die unternehmensinterne Verwaltung der Telearbeitsdienste in der Registry wird eine einfache Wartung ermöglicht, da die bereitgestellten Dienste unabhängig vom genutzten System am Telearbeitsort sind. Der Administrator hat die Möglichkeit neue Telearbeitsdienste zentral bereitzustellen und kann fehlerhafte Dienste für den Nutzer transparent austauschen.

Sicherheit

Durch den verwendeten Thin-Client-Ansatz wird die Speicherung unternehmensinterner Daten auf dem Telearbeitsrechner vermieden. Ein Zugriff auf die Arbeitsumgebung wird nur autorisierten Nutzern gewährt. Die Authentifikation und Verwaltung der Nutzer wird vom Portalframework übernommen. Heutige Portallösungen bieten ein sehr feingranulares und sicheres Rollen- und Benutzermanagement. Die Sicherheit der Arbeitsabläufe und Geschäftsprozesse kann durch die Integration entsprechender Sicherheitsdienste in die BPEL-Process-Flows gewährleistet werden. Außerdem können eine Reihe weiterer standardisierter Sicherheitsmechanismen für Web Services genutzt werden.

6.3.2 Bewertung

Im vorigen Abschnitt konnte gezeigt werden, dass die entwickelte Lösung die gestellten Anforderungen voll und ganz erfüllt. Gegenüber herkömmlichen Integrationsansätzen zeichnet sich die Lösung durch eine flexible Konfigurierbarkeit der Arbeitsumgebung, eine einfache Einbindung heterogener Dienste und die Personalisier- und Adaptierbarkeit der Benutzeroberfläche aus.

Die Lösung ermöglicht einen integrierten Zugriff auf alle Funktionen, die zur Bearbeitung der Aufgabenstellung eines Mitarbeiters erforderlich sind. Dies erlaubt eine zielgruppenspezifische und strukturierte Zusammenführung von bis dahin papiergebundenen oder in unterschiedlichen Systemen hinterlegten Informationen und Diensten und verhindert die Entstehung von Medienbrüchen zwischen den Anwendungen. Der verwendete Web-Service-Ansatz bietet eine einfache Integration der vorhandenen IT-Landschaft über standardisierte Schnittstellen. Bisher isolierte Legacy-Anwendungen können über Web-

Service-Gateways in die Telearbeitsumgebung integriert werden und damit auch die Geschäftsprozesse des Unternehmens verbessert werden.

Durch den Einsatz einer SOA-Infrastruktur können Geschäftsprozesse und Arbeitsabläufe permanent und zügig verbessert bzw. angepasst werden. Ändern sich die Prozesse und Abläufe, kann die IT-Abteilung des Unternehmens die Telearbeitsdienste zentral anpassen und für den Nutzer weitestgehend transparent umstellen. Damit wird dem Lebenszyklus von Geschäftsprozessen Rechnung getragen und die Flexibilität der Anwendungsintegration erhöht, da auf Prozessänderungen unmittelbar reagiert werden kann.

Die Wiederverwendung der Telearbeitsbasisdienste über mehrere Kanäle und Nutzergruppen hinweg führt zu kürzeren Entwicklungszeiten und Kostenvorteilen. Durch die wiederholte Nutzung und Verbesserung der einzelnen Komponenten steigt die Qualität der verwendeten Dienste.

Die Teamarbeit wird durch eine geschäftsprozessorientierte Aufgabenverteilung und die Bereitstellung von Funktionen zur Unterstützung synchroner und asynchroner Kommunikation auch innerhalb des Unternehmens verbessert.

Auf die Arbeitsumgebung kann über einen Webbrowser zugegriffen werden, was dem gängigen Thin-Client-Ansatz entspricht. Dadurch können vor allem Wartungskosten gespart werden, da eine Installation und Wartung von Anwendungsprogrammen auf dem Telearbeitsrechner entfallen kann. Außerdem wird durch den Thin-Client-Ansatz eine erhöhte Sicherheit gewährleistet, da keine sicherheitsrelevanten Daten auf den Telearbeitsrechner geladen werden müssen und der Zugriff nur nach Authentifizierung erfolgen kann.

Im Idealfall kann durch die Nutzung der vorgeschlagenen Lösung eine Investitionseinsparung ermöglicht werden. Dazu können Arbeitgeber und -nehmer die Übereinkunft treffen, dass der private PC des Arbeitnehmers als Telearbeitsplatz genutzt wird und der Arbeitgeber nur Abnutzungs- und Kommunikationskosten erstattet. Dies würde beiden Seiten entgegenkommen. Der Arbeitgeber spart Kosten und kann dank Thin-Client-Ansatz dennoch alle notwendigen Sicherheitsanforderungen erfüllen. Der Arbeitnehmer spart hingegen Platz für einen eventuell zusätzlichen Arbeitsrechner und kann ebenso die Kosten für den Netzzugang sparen. Da keine Daten auf dem eigenen Rechner gespeichert werden müssen, entfallen zusätzliche Sicherheitsmaßnahmen auf Benutzerseite. Somit kann der Rechner des Telearbeiters weiterhin von Familienmitgliedern genutzt werden.

Der Schulungsaufwand bei der Einführung von Telearbeit verringert sich, da die Telearbeiter über den Webbrowser auf die ihnen bekannte Arbeitsumgebung und auf die verschiedenen Backend-Systeme zugreifen können, ohne sich dabei mit den Benutzungskonzepten der einzelnen Systeme und der Integration direkt auseinander setzen zu müssen.

Die Arbeitsumgebung kann flexibel an die Bedürfnisse der Telearbeiter angepasst werden. Durch die Adaption an verschiedene Endgeräte und Modalitäten können Barrieren herkömmlicher Benutzeroberflächen überwunden werden und auch Benutzergruppen wie Behinderte in die Arbeitsabläufe im Unternehmen integriert werden.

Zusammenfassend kann man feststellen, dass die vorgeschlagene Lösung eine Reihe von Vorteilen gegenüber herkömmlichen Werkzeugen zur Unterstützung von Telearbeit bietet und darüber hinaus auch für andere Anwendungsgebiete wie B2B oder B2C geeignet ist.

KAPITEL 7
ZUSAMMENFASSUNG UND AUSBLICK

Das Große kommt nicht allein durch Impuls zustande, sondern ist eine Aneinanderkettung kleiner Dinge, die zu einem Ganzen vereint worden sind.

Vincent van Gogh (1853 - 1890)

7.1 Ergebnisse der Arbeit

Im Rahmen dieser Dissertation wurde ein integriertes Gesamtkonzept einer universellen Telearbeitsumgebung entwickelt und die praktische Umsetzung auf Basis einer serviceorientierten Architektur untersucht. Dazu wurden zu Beginn der vorliegenden Arbeit drei Thesen aufgestellt:

1. Mit der Umsetzung einer serviceorientierten Architektur auf Basis von Web Services lässt sich eine Telearbeitsumgebung schaffen, die im Vergleich zu vorhandenen Ansätzen flexibler konfigurierbar und damit universeller einsetzbar ist.
2. Komplexe Geschäftsprozesse und Arbeitsabläufe können durch die Komposition von Telearbeitsdiensten aus wiederverwendbaren Basisdiensten abgebildet werden. Dies führt zu kürzeren Entwicklungszeiten und Kostenvorteilen.
3. Mit Hilfe der generischen Beschreibung der Benutzerschnittstellen von Web Services kann eine flexible Integration der Telearbeitsdienste in ein Portal erfolgen und dem Telearbeiter damit eine einheitliche Benutzeroberfläche für alle benötigten Dienste und Anwendungen bereitgestellt werden.

In Kapitel 2 wurde der Begriff „Telearbeit“ eingegrenzt und definiert. Telearbeit wird vor allem durch die räumliche Trennung des Arbeitsplatzes vom Standort des Unternehmens und die Nutzung von elektronischen Medien zur Überbrückung dieser Trennung charakterisiert. Weiterhin wurden die vielfältigen Ausprägungen von Telearbeit dargestellt und Vor- und Nachteile dieser innovativen Arbeitsform aufgezeigt. Anschließend wurden die Probleme der bisherigen technischen Unterstützung von Telearbeit analysiert, um die Motivation für diese Dissertation zu verdeutlichen. Telearbeit kann nur dann erfolgreich sein, wenn Telearbeiter Zugriff auf die gleichen Anwendungen und Daten wie Büroarbeiter haben. Es muss folglich ein sicherer und reibungsloser Zugang zum Unternehmensnetz, E-Mail-Systemen, Kollaborationsumgebungen und Legacy-Anwendungen gewährleistet werden.

Um diese Probleme lösen und die Vielfalt der Anwendungsszenarien durch eine einzige Software-Lösung unterstützen zu können, wurde nach Technologien und Ansätzen zur flexiblen Umsetzung einer universellen Telearbeitsumgebung gesucht. Dazu wurden zuerst die Anforderungen an eine solche universelle Lösung spezifiziert. Darauf aufbauend wurden vorhandene Ansätze und Technologien zur praktischen Unterstützung von Telearbeit analysiert und bewertet. Diese wurden entsprechend ihrer Modelle und Herangehensweisen wie folgt klassifiziert:

- Anwendungen zur Unterstützung entfernten Arbeitens,
- Systeme zur Unterstützung kooperativen Arbeitens,
- Integrationslösungen.

Anwendungen zur Unterstützung entfernten Arbeitens erlauben den Fernzugriff auf Daten und Ressourcen und die Fernsteuerung entfernter Systeme. Systeme zur Unterstützung kooperativen Arbeitens sind vor allem für die Zusammenarbeit in Teams geeignet und ermöglichen die Kommunikation und Kollaboration der beteiligten Gruppenmitglieder. Integrationslösungen dienen der Zusammenführung verschiedenster Anwendungen teilweise auch über Unternehmensgrenzen hinweg und vorfolgen dabei entweder einen benutzerorientierten oder einen prozessorientierten Integrationsansatz.

Dabei konnten einige Defizite bei den untersuchten Technologien festgestellt werden, der wichtigste Nachteil ist aber, dass sie nicht den geforderten Grad an Flexibilität und Anpassungsfähigkeit bieten. Alle vorgestellten Lösungen bilden einen Teil der benötigten Funktionalität für Telearbeit ab, weshalb die Kombination mehrere Ansätze in einer ganzheitlichen Lösung vorgeschlagen wurde.

Um die Produktivität der Telearbeiter zu erhöhen, müssen die Aktivitäten der Mitarbeiter mit den IT-Systemen in Einklang gebracht werden. Die Lösung liegt in der Kombination von benutzerorientierter und prozessorientierter Integration der verschiedenen Anwendungssysteme in eine einheitliche Arbeitsumgebung. Eines der Alleinstellungsmerkmale serviceorientierter Architekturen auf der Basis von Web Services ist die Einführung einer prozessorientierten Schicht oberhalb der bisherigen, eher Middleware-zentrischen Ansätze. Deshalb wurde im Ergebnis dieser Arbeit eine serviceorientierte Architektur für Telearbeitsanwendungen entwickelt und deren Umsetzung detailliert beschrieben. Aufbauend auf die in Kapitel 4 ausführlich beschriebenen Technologien zur Umsetzung einer SOA wurde eine Infrastruktur zur Realisierung einer universellen Telearbeitsumgebung entwickelt und prototypisch implementiert. Dabei wurden die Technologien der Web Services und der Portalframeworks zugrunde gelegt.

Auf der Basis von Web Services lässt sich eine solche Telearbeitsumgebung schaffen, die im Vergleich mit vorhandenen Ansätzen universeller einsetzbar ist. Web Services bieten die Möglichkeit, heterogene Dienste und Anwendungen plattformunabhängig in einer webbasierten Umgebung zu integrieren [ACK+04]. Ebenso lassen sich bisher isolierte Dienste zu netzweiten Workflows komponieren. Das eröffnet neue Möglichkeiten nicht nur für Telearbeit sondern auch für B2B- und B2C-Anwendungen. Prinzipiell lassen sich die mit Web Services verwendeten Technologien auch zur unternehmensinternen Integration vorhandener Applikationen (EAI) nutzen.

Im Gegensatz zu vorhandenen, meist über Middleware durchgeführten punktuellen Integrationslösungen wird eine universelle Lösung präsentiert, welche vorhandene Legacy-Systeme, standardisierte Softwarekomponenten sowie eigene Neuentwicklungen gemeinsam integriert. Die Integration aller benötigten Anwendungen setzt dabei auf mehreren Ebenen an und bindet die für die einzelnen Schichten bereits existierenden Standards in ein gemeinsames Gedankenmodell. Die vorgestellte Architektur besteht dabei aus folgenden vier Integrationsebenen (von oben nach unten):

- die benutzerorientierte Integration aller Applikationen in einem Workplace,
- die Prozessintegration zur Abbildung der Geschäftsprozesse und Arbeitsabläufe,
- die Anwendungsintegration aufbauend auf Web Services
- und auf unterster Ebene die Datenintegration.

Ziel der Zusammenführung dieser verschiedenen Integrationstechnologien ist es, die benötigten Anwendungen flexibel zu Prozessen verbinden zu können und sie in einem adaptierbaren und personalisierbaren Arbeitsplatzportal zur Verfügung zu stellen. Dabei verfolgt der Ansatz ein zweistufiges Entwicklungsparadigma: Zuerst werden einzelne Basisdienste als Web Services umgesetzt („*programming in the small*“) und in einer zweiten Stufe zu

komplexen Prozessen zusammengefügt („*programming in the large*“). Jeder Mitarbeiter sieht dann in seiner Arbeitsumgebung genau die für ihn relevanten Anwendungskomponenten und deren Zusammenspiel in Geschäftsprozessen und Arbeitsabläufen und kann diese entsprechend flexibel an seine Bedürfnisse anpassen.

Die prozessorientierte Integration wird dabei entweder durch einen zentralen *Teleworking Service Integrator* oder bereits auf Seite des Service Providers durchgeführt. Im Rahmen dieser Arbeit wurde ein Framework entwickelt, das die flexible Komposition von Telearbeitsdiensten aus verschiedenen Basisdiensten unterstützt. Dabei werden semantische Annotationen genutzt, um die Suche nach geeigneten Basisdiensten zu vereinfachen. Die entwickelte Telearbeitsdienstontologie bildet dabei die in Kapitel 5.2 durchgeführte Klassifizierung auf eine Taxonomie für Basisdienste ab. Die geforderte Flexibilität zur dynamischen Auswahl von Web Services zur Laufzeit wird durch die Nutzung von abstrakten BPEL-Prozessen ermöglicht.

Da Portale sich durch die Unterstützung des serviceorientierten Ansatzes auszeichnen, führt an Ihnen als Frontend für verteilte heterogene Anwendungslandschaften künftig kein Weg mehr vorbei. Portale nehmen daher in einer SOA eine bedeutende Rolle ein, bilden sie doch den zentralen Einstiegspunkt in die flexible Architektur. Durch Nutzung der Portaltechnologie im Rahmen dieser Arbeit können die verschiedenen Anwendungssysteme zu einem einheitlichen, leicht erlernbaren Benutzerinterface zusammengefasst werden, was die dritte in dieser Arbeit aufgestellte These belegt. Je nach Rolle und aktueller Aufgabe des Telearbeiters wird eine eigens zusammengestellte Oberfläche angezeigt, die nur die jeweils benötigten Informationen und Funktionen enthält. Die Arbeitsumgebung kann durch den Telearbeiter selbst flexibel konfiguriert und angepasst werden.

Im Rahmen der vorliegenden Dissertation wurde ein weiteres Framework entwickelt, das die Suche und Auswahl geeigneter Telearbeitsdienste und die flexible Einbindung in das Portal ermöglicht. Zur automatisierten Einbindung der Dienste wurde ein Portlet entwickelt, das als Container für den Web-Service-Aufruf fungiert. Die damit verbundene Problemstellung der Generierung und Beschreibung von Benutzeroberflächen für Web Services bilden den zweiten Kernbereich der Arbeit. Die vorgestellte Lösung erlaubt es, die Schnittstellenbeschreibungen der Dienste (WSDL) mit zusätzlichen Tags zur generischen Beschreibung der Oberfläche zu versehen, welche dann zur Laufzeit geparst werden. Dabei wurde das an der University of Stanford entwickelte WSGUI-Konzept [KKM03] genutzt und weiterentwickelt. Die möglichst generische Beschreibung einer Benutzeroberfläche von Web Services ermöglicht außerdem eine bessere Anpassung an verschiedene Endgeräte und Modalitäten. Mit der prototypischen Implementierung der flexiblen Dienstintegration in das Portal konnte die dritte These bestätigt werden.

Eine wesentliche Kostenreduktion gegenüber herkömmlichen Integrationslösungen entsteht durch die Wiederverwendung, Standardisierung und Vereinfachung der Integration unterschiedlichster Anwendungssysteme, womit die zweite These belegt werden konnte. Dass sich hinter den Diensten eine heterogene Anwendungslandschaft mit unterschiedlichsten Zugriffsmechanismen und Plattformen verbirgt, bleibt für den Dienstnutzer transparent, er sieht nur die standardisierte Schnittstelle der Services.

Um die Vorteile von Integrationslösungen auf Basis von Portaltechnologien auch einmal zahlenmäßig zu beschreiben, seien hier die Ergebnisse der Analysten der Butler Group zitiert [Str05]. Neben dem geschäftlichen Nutzen wie bspw. dem Schaffen komplett neuer Geschäftsprozesse ergeben sich folgende technische Benefits: So stellen die Experten eine Reduzierung der Schnittstellen gegenüber herkömmlichen Middleware-Ansätzen um bis zu 50 Prozent in Aussicht und prognostizieren damit eine Einsparung für die Entwicklung der Schnittstellen um 25 bis 50 Prozent. Die Ausgaben für die Wartung sollen sogar um bis zu

75 Prozent gesenkt werden. Auch die Kosten, die mit dem Customizing der verschiedenen Anwendungsinterfaces verbunden sind, könnten um die Hälfte gesenkt werden.

Zusammenfassend kann man feststellen, dass sich die im Rahmen dieser Dissertation entwickelte Lösung gegenüber herkömmlichen Ansätzen zur Unterstützung von Telearbeit durch eine flexible Konfigurierbarkeit der Arbeitsumgebung, eine einfache Einbindung heterogener Dienste und die Personalisier- und Adaptierbarkeit der Benutzeroberfläche auszeichnet. Damit wird die These belegt, dass sich auf der Basis von Web Services eine Telearbeitsumgebung schaffen lässt, die im Vergleich mit vorhandenen Ansätzen flexibler und universeller einsetzbar ist.

7.2 Ausblick

7.2.1 Ausbaumöglichkeiten der vorgeschlagenen Lösung

Neben den präsentierten Ergebnissen der Arbeit wurden weitere Ausbaumöglichkeiten der Lösung und Ansatzpunkte für vertiefende Forschungen isoliert, die im Folgenden kurz vorgestellt werden sollen.

Im Rahmen dieser Arbeit wurden 2 unabhängige Frameworks zur flexiblen prozess- und benutzerorientierten Integration der Telearbeitsdienste vorgestellt. Diese Trennung hat den Vorteil, dass die prozessorientierte Komposition komplexer Telearbeitsdienste auch schon auf der Seite des Service Providers stattfinden kann. Ist der Dienstanbieter aber das eigene Unternehmen des Telearbeiters oder will der Telearbeiter selbst Dienste komponieren, wäre ein durchgängiger ganzheitlicher Entwicklungsprozess wünschenswert. Um dies zu ermöglichen, sollten Wege gesucht werden, die beiden Frameworks zu vereinen. Probleme ergeben sich dabei vor allem durch den unterschiedlichen Erfahrungsstand der einzelnen Nutzergruppen und dem sich daraus ergebenden Abstraktionsgrad bei der Entwicklung. Teile der Prozessintegration müssten dazu noch weiter vereinfacht und abstrahiert und vor allem durch graphische Tools unterstützt werden. Dazu können die semantischen Beschreibungen der Dienste und deren Zusammenspiel beitragen. An dieser Stelle wäre auch zu prüfen, inwieweit klassische Workflow-Systeme bei der Erstellung der abstrakten BPEL-Prozesse genutzt werden können.

Die Integration der Web Services in die Portalumgebung konnte prototypisch gezeigt werden. Die dabei zur Beschreibung der Benutzeroberfläche der Web Services genutzte WSGUI-Spezifikation muss noch weiter ausgebaut werden, um auch komplexere Benutzeroberflächen beschreiben zu können. Dabei sollten die Möglichkeiten der Integration moderner Standards wie XForms oder XUL und XAML geprüft werden. Wichtig ist auch die Unterstützung der Entwickler von Web Services bei der GUI-Beschreibung durch entsprechende Autorenwerkzeuge und graphische Editoren. Ein weiterer sehr interessanter Forschungsaspekt ist die Frage, wie die verschiedenen Oberflächen der einzelnen Web Services bei der prozessorientierten Komposition zusammengeführt werden können, um eine einheitliche Benutzeroberfläche für den kompositen Service zu erstellen.

Die möglichst generische Beschreibung einer Benutzeroberfläche von Web Services ermöglicht auch eine bessere Anpassung an verschiedene Endgeräte und Modalitäten. Bisherige Portalframeworks unterstützen nur eine kleine Menge von Adaptionsmechanismen für die Anpassung an verschiedene Endgeräte und Modalitäten wie z.B. WML oder VoiceXML. Um eine wirklich flexible und universelle Arbeitsumgebung zu schaffen, ist die Einbeziehung von Technologien aus dem Umfeld des Mobile und Ubiquitous Computing notwendig. Im Szenario mobiler Telearbeit ergeben sich spezielle Anforderungen an die Geräte- und Ortsunabhängigkeit der Arbeitsumgebung. Dabei ist es wichtig, dass der Mitarbeiter die bereitgestellten Dienste auf verschiedenen Endgeräten nutzen und

zwischen diesen auch dynamisch wechseln kann. Daraus ergibt sich die Notwendigkeit, dass die Lösung sich entsprechend an das Gerät anpassen kann (Adaption), wobei die Kontextinformationen und Profile des Mitarbeiters aber immer erhalten bleiben müssen. Ein ebenfalls noch offener Forschungspunkt, der bei der Nutzung mobiler Netze betrachtet werden muss, ist die Frage, wie sich das System verhält, wenn die Verbindung zum Telearbeiter abbricht und er für eine kurze Zeit offline ist. Deshalb sind im Anschluss an das Promotionsvorhaben vertiefende Forschungen zur dynamischen Adaption von Portalanwendungen an verschiedene stationäre und mobile Endgeräte unter Berücksichtigung des Anwendungskontextes geplant.

7.2.2 Weitere Entwicklung von Telearbeit

Mittlerweile wird Telearbeit nicht nur in Unternehmen der IT- und Telekommunikationsbranche, sondern ebenso im Versicherungs- und Bankenbereich, in Automobilunternehmen oder der öffentlichen Verwaltung durchgeführt. Mit der zunehmenden Verbreitung dieser Organisationsform ist auch ihre Akzeptanz gestiegen. Wissenschaftliche Begleitforschungen (unter anderem im Projekt TWIST[66]) zeigen, dass sich anfängliche Befürchtungen der Telearbeiter wie geringere Karrierechancen, Imageprobleme, Selbstausbeutung und eingeschränkte Kommunikation und Information nicht bestätigt haben.

Trotzdem wird das Potenzial an möglichen Telearbeitsplätzen bisher kaum ausgeschöpft. Die Unternehmen öffnen zwar zusehends ihre Computersysteme und Unternehmensnetze für den Fernzugang (*remote access*), aber zögern weiterhin, Mitarbeitern zu gestatten, ganze Tage von zu Hause zu arbeiten. Die Diffusion der Technik vollzieht sich folglich schneller, als die Führungsstile in den Unternehmen sich anpassen [Kor02]. Deshalb gibt es heute zwar viele Telearbeitsplätze im technischen Sinn, hingegen aber relativ wenig abhängig Beschäftigte, die alternierend oder permanent zu Hause arbeiten. Telearbeit stellt hohe Anforderungen an den Telearbeiter, die Kollegen und die Führungskräfte. Nur hohe Selbstständigkeit, Eigenverantwortung und Kommunikationsfähigkeit aller Beteiligten sichern die Einbindung der Mitarbeiter trotz zeitweiser räumlicher Trennung. Waren Manager früher eher skeptisch gegenüber Heim- bzw. Telearbeit und hatten die Befürchtung, die Produktivität könnte dabei sinken, so hat sich das Gegenteil herausgestellt. Bei Nortel [Nor05] ergab eine Befragung, dass über 94 % der Telearbeiter eine Produktivitätssteigerung von 15 bis 20 % verzeichnen können.

Besonders groß ist das unausgeschöpfte Potenzial in Deutschland. Hier ist der Unterschied zwischen der gegenwärtigen Praxis von permanenter und alternierender Telearbeit (mit 16,6% aller Erwerbstätigen) und dem Anteil an Arbeitsplätzen (mit 35,8%), die aufgrund ihres Tätigkeitsfeldes grundsätzlich dafür in Frage kämen, besonders hoch [Kor02]. Das Interesse der Arbeitnehmer ist sogar noch weitaus höher. Im Rahmen der Studie „Deutschland Online“[67] (2002) ergab eine Befragung über die Bereitschaft zur Telearbeit, dass über 60 % der Beschäftigten in nicht geringem Maße zur Ausübung ihrer Arbeit an einem dezentralen Arbeitsplatz bereit wären. Das Potenzial für Telearbeit ist somit nach wie vor erheblich größer als die derzeitige Praxis.

Doch mit den technischen Möglichkeiten steigt auch die Akzeptanz moderner Arbeitsformen weiter und immer mehr Mitarbeiter werden zumindest einen Tag pro Woche von zu Hause oder mobil arbeiten. Mit zunehmender Verbreitung von Breitband-Internetzugängen

[66] Vgl. http://www.twist.bmw.de/

[67] Vgl. http://www.studie-deutschland-online.de/

werden die Voraussetzungen für die produktive Nutzung von Telearbeit immer besser. Einige Branchenanalysten, wie z.B. Gartner [Jon04], bezeichnen Telearbeit sogar als „stille Revolution“, da eine sukzessive Durchdringung unserer Arbeitswelt mit neuen Technologien und Arbeitsformen stattfindet.

7.2.3 Diffusion und Weiterentwicklung des SOA- und Web-Service-Ansatzes

Serviceorientierte Architekturen bilden die unabdingbare Voraussetzung für dynamische, flexible IT-Systeme, die sich gleichermaßen im Fluss befinden. Unterliegt das Zusammenspiel der benötigten Anwendungen und deren Zusammensetzung in der Regel starker Dynamik, ist die Nutzung von standardisierten Schnittstellen zur losen Koppelung essentiell. Die Verwendung standardisierter Schnittstellen eröffnet wiederum neue Geschäftsmöglichkeiten und die Komplexität der Geschäftprozesse steigt. SOA bietet somit gegenüber herkömmlichen Integrationslösungen eine Verbesserung und Ausweitung der Geschäftsprozesse und eine Wertsteigerung durch die Integration vorher isolierter, heterogener Anwendungen (siehe Abbildung 7-1). Clients und Anwendungen können dadurch überall und jederzeit über standardisierte Schnittstellen kooperieren, ohne sich um Implementierungsdetails des Kommunikationspartners kümmern zu müssen. Somit können wirkliche plattform- und programmiersprachen-unabhängige Lösungen geschaffen werden. Durch die lose Koppelung kann flexibel auf Veränderungen der Anforderungen oder den Ausfall einzelner Komponenten reagiert werden.

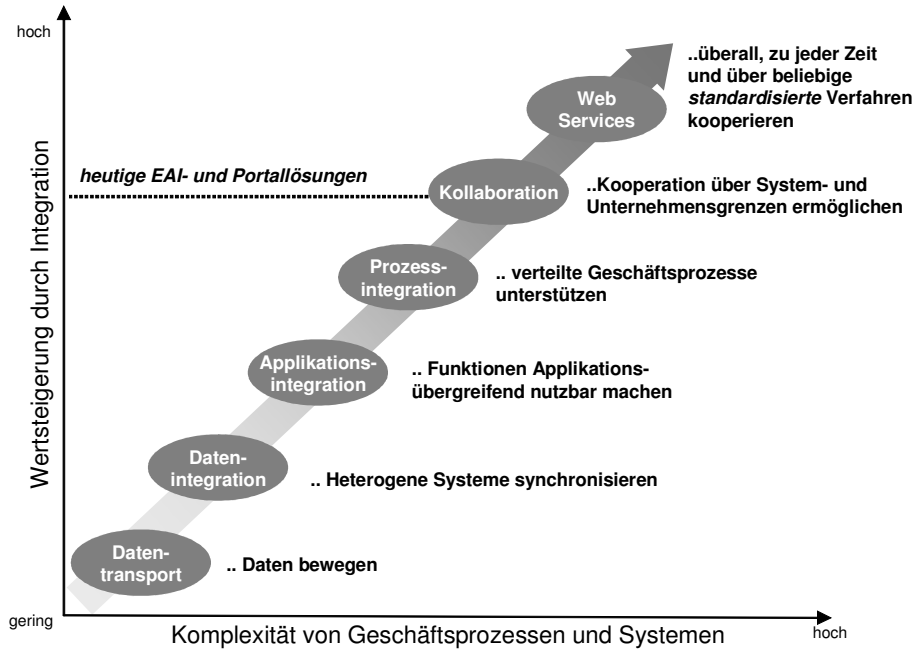

Abbildung 7-1: Entwicklungsstufen von Integrationslösungen (nach [BSZ01])

Web-Service-Technologien haben sich inzwischen in der Industrie einen festen Platz zur Lösung verschiedenster Integrationsprobleme erobert. Hierbei lösen Ansätze auf Basis XML-codierter entfernter Funktionsaufrufe und asynchron versandter Dokumente bereits sukzessive althergebrachte Middlewarearchitekturen ab. Bereits heute unterstützen alle auf J2EE bzw. .NET beruhenden Produkte die Nutzung von Web Services. Damit wurde der Grundstein dafür gelegt, dass in den nächsten Jahren die Entwicklung zu mächtigeren, auf SOA basierenden Anwendungen fortgesetzt wird, vorausgesetzt, dass die Verfügbarkeit von einzelnen Web Services der verschiedensten Service Provider weiter stetig wächst

[Jec03]. Dazu müssen sich in nächster Zeit erst einmal neue Geschäftsmodelle für die Bereitstellung und Nutzung solcher Dienste entwickeln und auf dem Markt etablieren.

In der Praxis kommen Web Services derzeit vor allem in geschlossenen Anwendungsumgebungen wie Intranets oder Extranets vor, wo vorhandene und dem Entwickler bekannte Anwendungen integriert werden sollen, ohne dass dafür die diversen Registrierungs- und Suchdienste wie UDDI nötig sind. Hinderungsgrund einer flächendeckenden Einführung ist dabei die Tatsache, dass Web Services noch immer kein vollständiges Rahmenwerk zur Adressierung aller auftretenden Kommunikationserfordernisse bieten. Aspekte wie Transaktionen, Sicherheit, Quality of Service, Verfügbarkeit und Authentifizierung sind bisher nur ansatzweise geklärt. So erscheint die Hierarchie der verschiedenen aufeinander aufsetzenden, sich wechselseitig ergänzenden und mitunter sogar offen konkurrierenden Protokollspezifikationen und -vorschläge zunehmend unübersichtlich und noch nicht vollständig ausdifferenziert, was sicherlich auch zur bisher zögerlichen Adaption im kommerziellen Massenmarkt geführt hat.

LITERATURVERZEICHNIS

[Abe03] Abendroth, J.: *Entwurf und prototypische Realisierung eines Dokumenten Management Systems mit RDF-Metadatenverwaltung*. Technische Universität Ilmenau, Diplomarbeit, September 2003.

[ACK+04] Alonso, G., Casati, F., Kuno, H., Machiraju, V.: *Web services: concepts, architectures and applications*. Springer Verlag, Berlin, Heidelberg, 2004.

[Ama05] Amazon.com Inc.: *Amazon Web Services.* Download: 20.Januar 2005. http://Web Services.amazon.com

[Bal96] Balzert, H.: *Lehrbuch der Software-Technik*. Lehrbücher der Informatik. Spektrum Verlag, Heidelberg, Berlin, 1996.

[BaS00] Back, A., Seufert, A.: *Computer Supported Cooperative Work (CSCW) - State-of-the-Art und zukünftige Herausforderungen.* In: CSCW - Workflow und Groupware, HMD - Praxis der Wirtschaftsinformatik, Heft 213, dpunkt.verlag, Heidelberg, 2000. S. 5-22.

[BBS00] Braun, I., Borcea, K., Schill, A.: *Working and learning at home - Designing a virtual working and learning environment for teleworkers*. In: Proceedings of 16th IFIP World Computer Congress 2000, International Conference on Educational Uses of Communication and Information Technologies, Beijing, China, 2000.

[BDH03] Büssing, A., Drodofsky, A., Hegendörfer, K.: *Telearbeit und Qualität des Arbeitslebens*. Hogrefe Verlag, Göttingen, 2003.

[BFH+99] Braun, I., Franze, K., Hess, R., Neumann, O., Schill, A.: *Integration von Telelearning- und Teleworking-Applikationen.* In: Engelien, M., Homann, J. (Hrsg.): Virtuelle Organisationen und neue Medien, Workshop GeNeMe99, Gemeinschaften in Neuen Medien, TU Dresden, Josepf Eul Verlag, Lohmar, Köln, 1999.

[BHS98a] Braun, I., Hess, R., Schill, A.: *Innovative Telearbeitsformen in klein- und mittelständischen Unternehmen*. In: Engelien, M., Bender, K. (Hrsg.): GeNeMe98, Gemeinschaften in Neuen Medien, TU Dresden, Josepf Eul Verlag, Lohmar, Köln, 1998.

[BHS98b] Braun, I., Hess, R., Schill, A.: *Teleworking support for small and medium-sized enterprises*. In: Proceedings of IFIP World Computer Congress '98, Wien, Budapest, 1998.

[BMB96] BMA, BMWI (Hrsg.): *Telearbeit – Chancen für neue Arbeitsformen, mehr Beschäftigung, flexible Arbeitszeiten*. Bonn, 1996.

[BPM01] Business Process Management Initiative: *Business Process Modeling Language (BPML) Specifikation*. BPMI.org, März 2001. http://www.bpmi.org/ BPML.htm

[BrS99] Braun, I., Schill, A.: *Experiences with regional teleworking support for small and medium-sized enterprises.* In: Proceedings of 1st European Regional Telematics Conference, Tanum, Schweden, 1999.

[BrS02] Braun, I., Schill, A.: *Building a universal teleworking environment using web services.* In: Proceedings of IASTED International Conference on Information and Knowledge Sharing (IKS 2002), St. Thomas, USA, 2002.

[BrS04] Braun, I., Schill, A.: *Der virtuelle Arbeitsplatz – Modell und Realisierung einer universellen Telearbeitsumgebung*; in Engelien, M., Meißner, K. (Hrsg.): Virtuelle Organisation und Neue Medien 2004 - Workshop GeNeMe2004, Reihe „Telekommunikation @ Medienwirtschaft", Band 16, Joseph Eul Verlag, Lohmar, Köln, 2004.

[BrS05] Braun, I., Schill, A.: *A service-oriented architecture for teleworking applications*. IASTED International Conference on Internet and Multimedia Systems and Applications (zur Veröffentlichung angenommen), Honolulu, August 2005.

[BrZ01] Braun, I., Zschuckelt, U.: *Designing a collaboration environment for teleworkers.* In: Proceedings of the World Conference on the WWW and Internet (WebNet) 2001, Orlando, USA, 2001.

[BSZ01] Braun, I., Schill, A., Zschuckelt, U.: *Virtuelle Mobilitätsdienste: Allgemeine Konzepte und Pilotvorhaben in der Region Dresden.* 18. Verkehrswissenschaftliche Tage Dresden - Verkehr und Mobilität in der Informations-gesellschaft; Dresden, 2001.

[Cer02] Cerami, E.: *Web services essentials.* O'Reilly, Tokyo, 2002.

[CGK+02] Curbera, Goland, Klein, Leymann, Roller, Thatte and Weerawarana: *Business Process Execution Language for Web Services Version 1.0*, Technical report, BEA Systems, International Business Machines Corporation, Microsoft Corporation, Juli 2002. ftp://www6.software.ibm.com/software/developer/library/ws-bpel1.pdf

[Cla03] Clark, K. G.: *In the Service of Cooperation*, Published on XML.com, O'Reilly Media Inc., 2003. http://www.xml.com/pub/a/ws/2003/07/08/ws-deviant.html

[Col95] Collardin, M.: *Aktuelle Rechtsfragen der Telearbeit.* Erich Schmidt-Verlag, Berlin, 1995.

[Col05] CollabWorx Inc.: *Integrated web collaboration solutions*, 2005. http://www.webwisdom.com

[Com05] *Communispace.* Communispace Corporation, Watertown, 2005. http://www.communispace.com

[Coy02] Coyle, F.P.: *XML, Web Services, and the Data Revolution*. Addison-Wesley, 2002.

[Cun01] Cunningham, M. J.: *B2B - erfolgreiche Geschäftsbeziehungen im Internet: Kosten senken.* Financial Times Prentice Hall, Amsterdam [u.a.], 2001.

[Deg05] Degenring, A.: *Enterprise Service Bus – Alle Wege führen nach Rom.* In: Java-Spektrum, Heft 4/2005, SIGS-Datacom, Troisdorf, 2005. S. 16-18.

[DLP+02] Dangelmaier, W.; Lessing, H.; Pape, U.; Rüther, M.: *Klassifikation von EAI-Systemen*. In: Meinhardt, S.; Popp, K. (Hrsg.) Enterprise-Portale & Enterprise Application Integration, HMD – Praxis der Wirtschaftsinformatik, Heft 225. dpunkt.verlag, Heidelberg, 2002. S. 61-71.

[DoJ04] Dostal, W., Jeckle, M.: *Semantik, Odem einer serviceorientierten Architektur.* In: Java Spektrum, Heft 1/2004, SIGS-Datacom, Troisdorf, 2004. S. 53-56.

[Dzi01] Dzida, W.: *Gebrauchstauglichkeit von Software. ErgoNorm: ein Verfahren zur Konformitätsprüfung von Software auf der Grundlage von DIN EN ISO 9241 Teile 10 und 11.* In: Bundesanstalt für Arbeitsschutz und Arbeitsmedizin (Hrsg.): Schriftenreihe der Bundesanstalt für Arbeitsschutz und Arbeitsmedizin: Forschung Fb 921. Wirtschaftsverlag NW, Verlag für Neue Wissenschaft, Bremerhaven, 2001.

[EbF03] Eberhart, A., Fischer, S.: *Web-Services. Grundlagen und praktische Umsetzung mit J2EE und.NET*. Hanser Verlag, München, Wien, 2003.

[EiL04] Eicker, S., Lelke, F.: *Grundzüge der Wirtschaftsinformatik*, Vorlesung WS 04/05, Universität Duisburg-Essen, 2004.

[eNo02] eNode Inc.: *Presentation and Integration of Web Services*, San Jose, 2002. http://www.enode.com/x/Web Services/

[EnS05] Endres, J., Siering, P. *Ferne Fenster. PC-Fernsteuerung nicht nur mit Remote Desktop.* In: c't - magazin für computertechnik, Nr. 10/2005, 2005. S. 96-100.

[ETO04] European Telework Online: *Telework and Telecommuting: Common Terms and Definitions.* Download: 18.August 2004. http://www.eto.org.uk/faq/faq02.htm

[EuR94] Europäischer Rat: *Europa und die globale Informationsgesellschaft. Empfehlungen für den Europäischen Rat* (Bangemann-Report). Brüssel, 1994.

[GaK02] Gareis, K. und Korte, W.B.: *Telework in Europe: Status Quo and Potenzial, Good Practices and Bad Practices.* In: Bundesanstalt für Arbeitsschutz und Arbeitsmedizin (Hrsg.): Telearbeit: Arbeits- und Gesundheitsschutz aus internationaler Sicht, Dortmund/ Berlin, 2002. S.43-74.

[Geb02] Gebauer, I.: *Auswirkungen häuslicher Telearbeit auf das Verkehrsverhalten und Aktionsräume - Eine Sekundäranalyse als explorative Studie.* Diskussionsbeiträge 12. Institut für Geographie der Universität Stuttgart, 2002.

[GI03] Gesellschaft für Informatik (GI): *Ankündigung des Symposiums „Entwicklung Web Service-basierter Anwendungen" im Rahmen der GI-Jahrestagung 2003*, Arbeitskreis „Web Services" der Gesellschaft für Informatik, 2003.

[GuÖ03] Gurzki, T., Özcan, N.: *Unternehmensportale: Kunden- Lieferanten- und Mitarbeiterportale in der betrieblichen Praxis.* Fraunhofer-IRB-Verlag, Stuttgart, 2003.

[HeB97] Hess, R., Braun, I.: *Mit Teleworking auf dem Weg in die Informationsgesellschaft.* In: Wirtschaftsdienst IHK Dresden 9/97, Dresden, 1997. S.20.

[Hec95] Heckl, H.: *Telearbeit aus Sicht der IT-Industrie.* In: HMD - Praxis der Wirtschaftsinformatik, Heft 185, dpunkt.verlag, Heidelberg, 1995.

[HeZ03] Hein, M., Zeller, H.: *Java Web Services*. Addisson-Wesley, München, 2003.

[Hof05] Hofmann, O.: *Analyse und Test verschiedener Ansätze zur Beschreibung und Komposition von komplexen Web Services*. Bachelorarbeit, TU Dresden, Februar 2005.

[Hoo94] Hoose, A.: *Telearbeit - Instrument zur Flexibilisierung und Deregulierung*. In: Arbeit und Sozialpolitik, 11-12/1994. S. 53-158.

[IAO97] Institut für Arbeitswirtschaft und Organisation (IAO): *Telearbeit auf dem Vormarsch.* Presseinformation Extra 09/97. IAO, Stuttgart, 1997.

[IBM00] IBM Web Services Architecture Team: *Web Services architecture overview*, 1.September 2000.
http://www-106.ibm.com/developerworks/Web Services/

[IBM04] IBM: *Web Services Transaction (WS-Transaction) Specification*. Technischer Bericht, November 2004.
http://www-128.ibm.com/developerworks/library/specification/ws-tx

[IBM05a] IBM: *Emerging Technologies Toolkit for Web Services and Autonomic Computing*. Technischer Bericht, Mai 2005.
http://www.alphaworks.ibm.com/tech/ettkws

[IBM05b] IBM: *Web Service Semantics - WSDL-S*. Technical Note, Version 1.0., April 2005.
http://www.alphaworks.ibm.com/g/g.nsf/img/semanticsdocs/$file/wssemantic_annotation.pdf

[Inf00] Infozone Group: *SchemoX (Schema Forms for XML).* 2000.
http://www.infozone-group.org/

[Jan03] Janssen, D.: *Kopplung von Web Services*. Fraunhofer IAO, Stuttgart, 2003.

[JäR01] Jäckel, M., Rövekamp, C.: *Alternierende Telearbeit.* Westdeutscher Verlag, Wiesbaden, 2001.

[JCP03] Java Community Process: *JSR 168: Portlet Specification*. Java Specification Request, Final Release, 27.Oktober 2003.
http://www.jcp.org/en/jsr/detail?id=168

[Jec03] Jeckle, M.: *Web Services und das Semantic Web*. Vortrag auf der Semantic Web Tour des Deutsch-Österreichischen W3C-Büros am 17.Juni 2003. München, 2003.
http://www.jeckle.de

[JMP03] Jablonski, S., Meiller, C., Petrov, I.: *Web-Services und Semantic Web.* In Web-Services, Hans-Peter Fröschle (Hrsg.), HMD - Praxis der Wirtschaftsinformatik, Heft 234, Seite 78–86, dpunkt.verlag, Heidelberg, 2003.

[Joh97] Johanning, D.: *Telearbeit. Einführung und Leitfaden für Unternehmer und Mitarbeiter*. Carl Hanser Verlag, München Wien, 1997.

[Jon04] Jones, C.: *Teleworking: The Quiet Revolution*. Gartner Forecast Analysis, 17. September 2004.

[Kir96] Kirchmair, G.: *Telearbeit. Realität und Zukunft. Telearbeit und Schlüsselqualifikationen in der postmodernen Wissensgesellschaft*. Verlag des ÖGB, Wien, 1996.

[KKM03] Kassoff, M., Kato, D., Mohsin, W.: *Creating GUIs for Web Services*. IEEE Internet Computing, September/Oktober 2003. Vol. 7, No. 4, 66-73.

[Knu02] Knuth, M.: *Web Services - Einführung und Übersicht.* Software & Support Verlag GmbH, Frankfurt, 2002.

[KoK98] Kordey, N., Korte, W. B.: *Telearbeit erfolgreich realisieren. Zielorientiertes Business computing*. Vieweg Verlag, Braunschweig; Wiesbaden, 1998.

[Kor98] Korte, W. B.: *Perspektiven von Telearbeit und Telekooperation in Wirtschaft und Verwaltung.* In: Godehardt, B., Korte, W., Michelsen, U., Quadt, H.-P. (Hrsg.): Management Handbuch Telearbeit, Hüthig Verlag, Heidelberg, 1998.

[Kor02] Kordey, N.: *Verbreitung der Telearbeit in 2002 - Internationaler Vergleich und Entwicklungstendenzen*. In: empirica Schriftenreihe: Telearbeit, Report 02/2002, Bonn, 2002.

[KOW01] Kreilkamp, P., Oldenburg, S., Wakiel, B.: *Telearbeit – Leitfaden für flexibles Arbeiten in der Praxis.* BMA, BMWI, BMBF (Hrsg.), Bonn, 2001.

[Mic97] Microsoft Corporation: *Distributed Component Object Model Protocol (DCOM): Architecture.* Technischer Bericht, Juli 1997. http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dndcom/html/msdn_dcomarch.asp

[MOZ05] Mozilla: *Implement W3C XForms in browser and composer*. 2005. https://bugzilla.mozilla.org/show_bug.cgi?id=97806

[NeL05] Newcomer, E., Lomow, G.: *Understanding SOA with Web Services*. Independent technology guides, Addison-Wesley, Boston [u.a.], 2005.

[Nil76] Nilles, J.: *The Telecommunications – Transport trade off*. Wiley, New York, 1976.

[Nil94] Nilles, J.: *Making Telecommuting Happen - A Guide for Telemanagers and Telecommuters.* Van Nostrand Reinhold, New York, 1994.

[Nor05] Nortel Networks: *Mit Teleworking zu mehr Mobilität - Über 10 Jahre Entwicklung bei Nortel*. 2005. http://www.nortel.com/corporate/global/emea/germany/collateral/nn110521-122004_de.pdf

[OAS04a] OASIS: *Web Services Security: SOAP Message Security 1.0 (WS-Security).* OASIS Standard, März 2004. http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0.pdf

[OAS04b] OASIS Web Services Reliable Messaging TC: *WS-Reliability 1.1*. OASIS Standard, 15. November 2004. http://docs.oasis-open.org/wsrm/ws-reliability/v1.1

[OAS05a] OASIS: *UDDI Specifications*. Download: 18.März 2005. http://www.oasis-open.org/committees/uddi-spec/doc/tcspecs.htm

[OAS05b] OASIS: *Web Services Business Process Execution Language Version 2.0.* Working Draft, 20.Mai 2005. http://www.oasis-open.org/committees/download.php/12791/wsbpel-specification-draft-May-20-2005.html

[Öst03] Österle, H. (Hrsg.): *Business Engineering : auf dem Weg zum Unternehmen des Informationszeitalters.* Springer, Berlin, Heidelberg, 2003.

[OIO05] Orientation in Objects GmbH: *REST Web Services - Eine Einführung*. Orientation in Objects GmbH, Mannheim, Download: 11.7.2005. http://www.oio.de/public/xml/rest-Web Services.htm

[OMG03] OMG: *The Common Object Request Broker Specification.* OMG Specification v3.0.2, OMG - Object Management Group, Februar 2003. http://www.omg.org/technology/documents/formal/corba_2.htm

[PEL03] Peltz, C.: *Web Service Orchestration*. Technical White Paper, Hewlett Packard, Januar 2003. http://devresource.hp.com/drc/technical_white_papers/WSOrch/ WSOrchestration.pdf

[POS92] IEEE: *Portable Operating System Interface for Computer Environments (POSIX);* IEEE Standard 10003.0 „Guide to POSIX open Systems Environment“, 1992.

[QuW03] Quatz, J., Wichmann, T.: *Basisreport Integration mit Web Services - Konzept, Fallstudien und Bewertung.* BERLECON RESEARCH GmbH, Berlin, 2003.

[Rau00] Rauscher, G.: *Die HP-Zeitphilosophie: Gestaltung der Lebensparameter Arbeit, Zeit und Geld.* In: Key. Nr. 4, Wintersemester 1999/2000. S. 38-39.

[ReG98] Rensmann, J.H., Gröpler, K.: *Telearbeit – Ein praktischer Wegweiser*. Springer, Berlin, Heidelberg, 1998.

[RFC94] IETF Network Working Group: T. Berners-Lee: *RFC 1630 - Universal Resource Identifiers in WWW*, 1994. http://www.ietf.org/rfc/rfc1630.txt

[RLB05] Ruth, D., Lorz, A., Braun, I.: *Webbasierte Groupware-Anwendungen für die Kooperation in verteilten Projektteams und virtuellen Unternehmen*. GeNeMe05-Workshop, Dresden, Oktober 2005.

[RMS+00] Reichwald, R., Möslein, K., Sachenbacher, H., Englberger, H.: *Telekooperation. Verteilte Arbeits- und Organisationsformen.* Springer Verlag, 2. Aufl., Berlin, 2000.

[ScS01] Scribner, K., Stiver, M.C.: *SOAP developer’s guide.* Markt+Technik Verlag, 2001.

[ScW02] Schelp, J., Winter, R.: *Enterprise Portals und Enterprise Application Integration – Begriffsbestimmung und Integrationskonzeptionen*. In: Meinhardt, S.; Popp, K. (Hrsg.): Enterprise-Portale & Enterprise Application Integration, HMD – Praxis der Wirtschaftsinformatik, Heft 225. dpunkt.verlag, Heidelberg, 2002. S. 6-20.

[SHB98] Schill, A., Hess, R., Braun, I.: *Innovative Telearbeitsformen in klein- und mittelständischen Unternehmen*. 4. Sächsisches Informatik-Kolloquium, Dresden, 1998.

[Spi05] Spillner, J.: *Web Services Graphical User Interface (WSGUI).* Standard Specification and Documentation, Revision 0.99.2, 1.Juli 2005. http://www.inf.tu-dresden.de/~js177634/webservices/

[SPS04] Srinivasan, N., Paolucci, M., Sycara, K.: *Adding OWL-S to UDDI, implementation and throughput*. First International Workshop on Semantic Web Services and Web Process Composition, San Diego, USA, 2004.

[STK02] Snell, J., Tidwell, D., Kulchenko, P.: *Web Service-Programmierung mit SOAP*. O'Reilly Verlag, 2002.

[Str05] Strehlitz, M.: *Am Portal führt kein Weg vorbei.* In: Computer Zeitung, Nr. 27, 4.Juli 2005, Konradin Verlagsgruppe, Leinfelden-Echterdingen, 2005.

[Sun03] SUN Microsystems Inc.: *Java Remote Method Invocation (RMÌ): Specification.* Technischer Bericht v1.4.2, Februar 2003.
http://www.omg.org/technology/documents/formal/corba_2.htm

[Tea05] *Teamspace - let's work together*, 5 POINT AG, Darmstadt, Download: 1.Februar 2005.
http://www.teamspace.de/

[TMF04] Thomas Manes, A., Fritsch, W.: *Java oder .Net?* In: InformationWeek, Heft 9-10/2004, CMP-WEKA-Verlag, Poing, 2004.

[Tof80] Toffler, A.: *The third wave.* Morrow, New York, 1980.

[TSM+95] Teufel, S., Sauter, C., Müller, T., Bauknecht, K.: *Computerunterstützung für die Gruppenarbeit.* Addison-Wesley, Bonn, 1995.

[UDD05] UDDI.org: *Universal Description, Discovery and Integration*, Offizielle Website, Download: 1.Februar 2005.
http://www.uddi.org

[W3C00] W3C: *Simple Object Access Protocol (SOAP) 1.1.* W3C Note 8.Mai 2000, Technischer Bericht, W3C, 2000.
http://www.w3.org/TR/2000/NOTE-SOAP-20000508/

[W3C01a] W3C: *XML Key Management Specification (XKMS).* W3C Note, 30.März 2001. Technischer Bericht, W3C, 2001.
http://www.w3.org/TR/2001/NOTE-xkms-20010330/

[W3C01b] W3C: *XML Signature - Syntax and Processing*, W3C Recommendation, 12.Februar 2002.
http://www.w3.org/TR/xmldsig-core/

[W3C02a] W3C: *Web Services Architecture Requirements*. W3C Working Draft, 11.Oktober 2002.
http://www.w3.org/TR/2002/WD-wsa-reqs-20021011/

[W3C02b] W3C: *Web Service Choreography Interface (WSCI) 1.0.* W3C Note, 8.August 2002.
http://www.w3.org/TR/wsci/

[W3C02c] W3C: *XML Encryption - Syntax and Processing*. W3C Recommendation, 10.Dezember 2002.
http://www.w3.org/TR/xmlenc-core/

[W3C03a] W3C: *Web Services Architecture Working Group Participants*. Technischer Bericht, August 2003.
http://www.w3.org/2002/ws/arch/#participants

[W3C03b] W3C: *XForms 1.0*. Recommendation, 14.Oktober 2003.
http://www.w3.org/TR/2003/REC-xforms-20031014/

[W3C04] W3C: *Web Services Glossary.* W3C Working Group Note, 11.Februar 2004.
http://www.w3.org/TR/ws-gloss

[W3C05] W3C: *Web Services Description Working Group*. Download: 27.Februar 2005.
http://www.w3.org/2002/ws/desc/

[Web03] Webfair AG (Hrsg.) *Volkswagen PartnerNet – Webbasiertes Informationsmanagement im Händlernetz*. München, 2003.
http://www.webfair.com

[Wed94] Wedde, P.: *Telearbeit. Handbuch für Arbeitnehmer, Betriebsräte und Anwender*. Bund-Verlag GmbH, Köln, 1994.

[WFM99] Workflow Management Coalition: *Terminology & Glossary*. Februar 1999.
http://www.wfmc.org/standards/docs/TC-1011_term_glossary_v3.pdf

[WHA05] Web Hypertext Application Technology Working Group: *Web Forms 2.0*. Working Draft, 10.Juni 2005.
http://whatwg.org/specs/web-forms/current-work/

[WIK05] Wikipedia: *Wireless LAN*. Download: 12.Juni 2005.
http://de.wikipedia.org/wiki/WLAN

[Win01] Winker, G. (Hrsg.): *Telearbeit und Lebensqualität: zur Vereinbarkeit von Beruf und Familie*. Campus-Verlag, New York, 2001.

[Wit03] Witte, H.: *Was bei der Einführung von Telearbeit zu beachten ist*. Computerwoche.de, 6.August 2003.
http://www.computerwoche.de/

[ZCL05] Zhou, C., Chia, N., Lee, B.-S.: *Web Services Discovery with DAML-QoS Ontology*. Idea Group, Nanyang Technological University, Singapore, 2005.

ABBILDUNGSVERZEICHNIS

TABELLENVERZEICHNIS

QUELLCODE-VERZEICHNIS

ABKÜRZUNGSVERZEICHNIS

ADSL	Asymmetric Digital Subscriber Line
API	Application Programming Interface
ASP	Active Server Pages
B2B	Business to Business
B2C	Business to Consumer
B2E	Business to Employer
BPEL(4WS)	Business Process Execution Language (for Web Services)
BPMI	Business Process Management Initiative
BPML	Business Process Modeling Language
BPMS	Business Process Management System
BPWS4J	Business Process Execution Language for Web Services for Java
BSCW	Basic Support for Cooperative Work
C2C	Consumer to Consumer
CA	Certification Authority
CAD	Computer Aided Design
CI	Customer Information
CIL	C Intermediate Language
CLR	Common Language Runtime
CMS	Content Management System
CORBA	Common Object Request Broker Architecture
CRM(S)	Customer Relationship Management (System)
CSCW	Computer Supported Cooperative Work
CSS	Cascading Style Sheet
DAML(-S)	DARPA Agent Markup Language (for Services)
DARPA	Defense Advanced Research Projects Agency
DCOM	Distributed Component Object Model
DES	Data Encryption Standard
DFKI	Deutsches Forschungszentrum für Künstliche Intelligenz
DMS	Document Management System
DSL	Digital Subscriber Line
DV	Datenverarbeitung
EAI	Enterprise Application Integration
ERP(S)	Enterprise Resource Planning (System)
ESB	Enterprise Service Bus

ETTK	Emerging Technologies Toolkit
FDD	Frequency Division Duplex
GIOP	General Inter-ORB Protocol
GMD	Gesellschaft für Mathematik und Datenverarbeitung
GPRS	General Packet Radio Service
GSM	Global System for Mobile Communications
GUI	Graphical User Interface
GUIDD	Graphical User Interface Deployment Descriptor
HAG	Heimarbeitsgesetz
HDSL	High-Data-Rate Digital Subscriber Line
HR(MS)	Human Resources (Management System)
HTML	Hyper-Text Markup Language
HTTP(S)	Hyper-Text Transfer Protocol (Secure)
I&K-...	Informations- und Kommunikations...
IDL	Interface Definition Language
IEEE	Institute of Electrical and Electronics Engineers
IETF	Internet Engineering Task Force
IIOP	Internet Inter-ORB Protocol
IIS	Internet Information Server
IL	Intermediate Language
IMAP	Internet Message Access Protocol
ISDN	Integrated Services Digital Network
IT	Informationstechnik
ITU	International Telecommunication Union
J2EE	Java 2 Enterprise Edition
JAXP	Java API for XML Processing
JAXB	Java Architecture for XML Binding
JAXR	Java API for XML Registries
JAX-RPC	Java API for XML-based RPC
JIT	Just in Time
JSR	Java Specification Request
LDAP	Lightweight Directory Access Protocol
MAPI	Messaging Application Programming Interface
MB	Mega Byte
MIME	Multipurpose Internet Mail Extensions
MOM	Message Oriented Middleware

MS	Microsoft
NDS	Novell Directory Services
OASIS	Organization for the Advancement of Structured Information Standards
ODBC	Open Data Base Connectivity
OLAP	Online Analytical Processing
OMG	Object Management Group
ORB	Object Request Broker
OS	Open Source
OSD	Open Search Description
OSF	Open Software Foundation
OWA	Outlook Web Access
OWL(-S)	Web Ontology Language (for Services)
PC	Personal Computer
PDA	Personal Digital Assistant
PKI	Public Key Infrastructure
POP3	Post Office Protocol Version 3
PR	Public Relations
QoS	Quality of Service
RDA	Remote Data Access
RDF(S)	Resource Description Framework (Schema)
RDP	Remote Desktop Protocol
RDQL	RDF Query Language
REST	REpresentational State Transfer
RMI	Remote Method Invocation
RPC	Remote Procedure Call
RSA	asymmetrisches Kryptosystem, nach Erfindern Rivest, Shamir und Adleman benannt
RSS	Really Simple Syndication
SAAJ	SOAP with Attachments API for Java
SCM(S)	Supply Chain Management (System)
SDK	Software Development Kit
SMS	Short Message Service
SMTP	Simple Mail Transfer Protocol
SOA	Service Oriented Architecture
SOAP	(Simple Object Access Protocol)
SOAP4J	SOAP for Java

SOHO	Small Office Home Office
SQL	Structured Query Language
S-UUDI	Simple UDDI
SWSL	Semantic Web Service Language
UBR	UDDI Business Registry
UDDI	Universal Description, Discovery and Integration
UDDI4J	Universal Description, Discovery and Integration for Java
UI	User Interface
UML	Unified Modeling Language
UMTS	Universal Mobile Telecommunications System
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
VoiceXML	Voice eXtended Markup Language
VoIP	Voice over IP (Internet Protocol)
VPN	Virtual Private Network
W3C	World Wide Web Consortium
WAP	Wireless Application Protocol
WAR	Web Application aRchive
WFMC	Workflow Management Coalition
WFMS	Workflow Management System
WHAT	Web Hypertext Application Technology Working Group
WLAN	Wireless Local Area Network
WML	Wireless Markup Language
WMS	Wissensmanagementsystem
WPAN	Wireless Personal Area Network
WS-BPEL	Web Services Business Process Execution Language
WSCI	Web Service Choreography Interface
WSCL	Web Service Coversation Language
WSDL	Web Service Description Language
WSDL-S	Web Service Semantics
WSDP	Web Service Developer Pack
WSEE	Web Services for J2EE
WSFL	Web Service Flow Language
WSGUI	Web Services Graphical User Interface
WSMF	Web Service Modeling Framework
WSRP	Web Service for Remote Portlets

WSUI	Web Services User Interface
WWW	World Wide Web
WYSIWYG	What You See Is What You Get
XAML	eXtensible Application Markup Language
XKMS	XML Key Management Specification
XML	eXtensible Markup Language
XPATH	XML Path Language
XSLT	eXtensible Stylesheet Language Transformation
XUL	XML User Interface Language

ANHANG

A.1 Schnittstellenbeschreibung der SimpleDocumentManager-Klasse

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions targetNamespace="http://documentManager.doc" x
 xmlns:impl="http://documentManager.doc" xmlns:intf="http://documentManager.doc"
 xmlns:apachesoap="http://xml.apache.org/xml-soap"
 xmlns:wsdlsoap="http://schemas.xmlsoap.org/wsdl/soap/"
 xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:tns4="http://types.axis.apache.org"
 xmlns:tns1="http://accessManager.sys" xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
 xmlns="http://schemas.xmlsoap.org/wsdl/">
 <wsdl:types>
  <schema xmlns="http://www.w3.org/2001/XMLSchema"
   targetNamespace="http://accessManager.sys">
   <import namespace="http://schemas.xmlsoap.org/soap/encoding/"/>
   <complexType abstract="true" name="Owner">
    <sequence>
     <element name="ownerId" type="xsd:int"/>
     <element name="title" nillable="true" type="xsd:string"/>
    </sequence>
   </complexType>
   <complexType name="Person">
    <complexContent>
     <extension base="tns1:Owner">
      <sequence>
       <element name="firstname" nillable="true" type="xsd:string"/>
       <element name="lastname" nillable="true" type="xsd:string"/>
      </sequence>
     </extension>
    </complexContent>
   </complexType>
  </schema>
  <schema xmlns="http://www.w3.org/2001/XMLSchema"
   targetNamespace="http://documentManager.doc">
   <import namespace="http://schemas.xmlsoap.org/soap/encoding/"/>
   <complexType name="Description">
    <sequence/>
   </complexType>
   <complexType name="Format">
    <sequence>
     <element name="formatID" type="xsd:int"/>
     <element name="formatType" nillable="true" type="xsd:string"/>
    </sequence>
   </complexType>
   <complexType name="Language">
    <sequence>
     <element name="languageCode" nillable="true" type="xsd:string"/>
     <element name="languageDictionary" nillable="true" type="xsd:anyType"/>
     <element name="languageId" type="xsd:int"/>
     <element name="languageName" nillable="true" type="xsd:string"/>
     <element name="languageParser" nillable="true" type="xsd:anyType"/>
    </sequence>
   </complexType>
   <complexType abstract="true" name="State">
    <sequence>
     <element name="state" nillable="true" type="xsd:anyType"/>
     <element name="stateId" type="xsd:int"/>
     <element name="stateName" nillable="true" type="xsd:string"/>
    </sequence>
   </complexType>
   <complexType name="Subject">
    <sequence/>
   </complexType>
   <complexType name="Type">
    <sequence>
     <element name="typeId" type="xsd:int"/>
     <element name="typeName" nillable="true" type="xsd:string"/>
    </sequence>
   </complexType>
   <complexType name="Document">
    <sequence>
     <element name="creationDate" nillable="true" type="xsd:dateTime"/>
```

```
     <element name="creator" nillable="true" type="tns1:Person"/>
     <element name="description" nillable="true" type="impl:Description"/>
     <element name="format" nillable="true" type="impl:Format"/>
     <element name="identifier" type="xsd:int"/>
     <element name="language" nillable="true" type="impl:Language"/>
     <element name="state" nillable="true" type="impl:State"/>
     <element name="subject" nillable="true" type="impl:Subject"/>
     <element name="title" nillable="true" type="xsd:string"/>
     <element name="type" nillable="true" type="impl:Type"/>
     <element name="updateDate" nillable="true" type="xsd:dateTime"/>
    </sequence>
   </complexType>
  </schema>
  <schema xmlns="http://www.w3.org/2001/XMLSchema"
   targetNamespace="http://types.axis.apache.org">
   <import namespace="http://schemas.xmlsoap.org/soap/encoding/"/>
   <complexType name="URI">
    <sequence>
     <element name="fragment" nillable="true" type="xsd:string"/>
     <element name="genericURI" type="xsd:boolean"/>
     <element name="host" nillable="true" type="xsd:string"/>
     <element name="path" nillable="true" type="xsd:string"/>
     <element name="port" type="xsd:int"/>
     <element name="queryString" nillable="true" type="xsd:string"/>
     <element name="scheme" nillable="true" type="xsd:string"/>
     <element name="schemeSpecificPart" nillable="true" type="xsd:string"/>
     <element name="userinfo" nillable="true" type="xsd:string"/>
    </sequence>
   </complexType>
  </schema>
 </wsdl:types>
   <wsdl:message name="addDocumentResponse">
      <wsdl:part name="addDocumentReturn" type="xsd:boolean"/>
   </wsdl:message>
   <wsdl:message name="storeDocumentResponse1">
      <wsdl:part name="storeDocumentReturn" type="xsd:boolean"/>
   </wsdl:message>
   <wsdl:message name="getAllDocumentsRequest">
   </wsdl:message>
   <wsdl:message name="getDocumentUriRequest">
      <wsdl:part name="documentID" type="xsd:int"/>
   </wsdl:message>
   <wsdl:message name="getDocumentInfoRequest">
      <wsdl:part name="documentID" type="xsd:int"/>
   </wsdl:message>
   <wsdl:message name="getDocumentInfoResponse">
      <wsdl:part name="getDocumentInfoReturn" type="impl:Document"/>
   </wsdl:message>
   <wsdl:message name="setDocumentStateRequest">
      <wsdl:part name="documentID" type="xsd:int"/>
      <wsdl:part name="stateID" type="xsd:int"/>
   </wsdl:message>
   <wsdl:message name="deleteDocumentResponse1">
      <wsdl:part name="deleteDocumentReturn" type="xsd:boolean"/>
   </wsdl:message>
   <wsdl:message name="getDocumentUriResponse">
      <wsdl:part name="getDocumentUriReturn" type="tns4:URI"/>
   </wsdl:message>
   <wsdl:message name="deleteDocumentRequest1">
      <wsdl:part name="document" type="impl:Document"/>
   </wsdl:message>
   <wsdl:message name="storeDocumentResponse">
      <wsdl:part name="storeDocumentReturn" type="xsd:boolean"/>
   </wsdl:message>
   <wsdl:message name="storeDocumentRequest">
      <wsdl:part name="document" type="impl:Document"/>
      <wsdl:part name="url" type="xsd:anyType"/>
   </wsdl:message>
   <wsdl:message name="getAllStateRequest">
   </wsdl:message>
   <wsdl:message name="loadDocumentResponse">
      <wsdl:part name="loadDocumentReturn" type="xsd:anyType"/>
   </wsdl:message>
   <wsdl:message name="setDocumentStateResponse">
      <wsdl:part name="setDocumentStateReturn" type="xsd:boolean"/>
   </wsdl:message>
   <wsdl:message name="getDocumentUriReadOnlyResponse">
      <wsdl:part name="getDocumentUriReadOnlyReturn" type="tns4:URI"/>
```

```
    </wsdl:message>
    <wsdl:message name="getDocumentUriReadOnlyRequest">
       <wsdl:part name="documentID" type="xsd:int"/>
    </wsdl:message>
    <wsdl:message name="addDocumentRequest">
       <wsdl:part name="document" type="impl:Document"/>
    </wsdl:message>
    <wsdl:message name="loadDocumentRequest">
       <wsdl:part name="documentID" type="xsd:int"/>
    </wsdl:message>
    <wsdl:message name="getAllDocumentsResponse">
       <wsdl:part name="getAllDocumentsReturn" type="soapenc:Array"/>
    </wsdl:message>
    <wsdl:message name="storeDocumentRequest1">
       <wsdl:part name="document" type="impl:Document"/>
       <wsdl:part name="file" type="xsd:anyType"/>
    </wsdl:message>
    <wsdl:message name="loadDocumentReadOnlyRequest">
       <wsdl:part name="documentID" type="xsd:int"/>
    </wsdl:message>
    <wsdl:message name="getAllStateResponse">
       <wsdl:part name="getAllStateReturn" type="soapenc:Array"/>
    </wsdl:message>
    <wsdl:message name="deleteDocumentRequest">
       <wsdl:part name="documentID" type="xsd:int"/>
    </wsdl:message>
    <wsdl:message name="loadDocumentReadOnlyResponse">
       <wsdl:part name="loadDocumentReadOnlyReturn" type="impl:FileOutputStream"/>
    </wsdl:message>
    <wsdl:message name="deleteDocumentResponse">
       <wsdl:part name="deleteDocumentReturn" type="xsd:boolean"/>
    </wsdl:message>
    <wsdl:portType name="SimpleDocumentManager">
       <wsdl:operation name="loadDocument" parameterOrder="documentID">
          <wsdl:input name="loadDocumentRequest" message="impl:loadDocumentRequest"/>
          <wsdl:output name="loadDocumentResponse" message="impl:loadDocumentResponse"/>
       </wsdl:operation>
       <wsdl:operation name="addDocument" parameterOrder="document">
          <wsdl:input name="addDocumentRequest" message="impl:addDocumentRequest"/>
          <wsdl:output name="addDocumentResponse" message="impl:addDocumentResponse"/>
       </wsdl:operation>
       <wsdl:operation name="storeDocument" parameterOrder="document url">
          <wsdl:input name="storeDocumentRequest" message="impl:storeDocumentRequest"/>
          <wsdl:output name="storeDocumentResponse" message="impl:storeDocumentResponse"/>
       </wsdl:operation>
       <wsdl:operation name="storeDocument" parameterOrder="document file">
          <wsdl:input name="storeDocumentRequest1" message="impl:storeDocumentRequest1"/>
          <wsdl:output name="storeDocumentResponse1"
           message="impl:storeDocumentResponse1"/>
       </wsdl:operation>
       <wsdl:operation name="loadDocumentReadOnly" parameterOrder="documentID">
          <wsdl:input name="loadDocumentReadOnlyRequest"
           message="impl:loadDocumentReadOnlyRequest"/>
          <wsdl:output name="loadDocumentReadOnlyResponse"
           message="impl:loadDocumentReadOnlyResponse"/>
       </wsdl:operation>
       <wsdl:operation name="deleteDocument" parameterOrder="documentID">
          <wsdl:input name="deleteDocumentRequest" message="impl:deleteDocumentRequest"/>
          <wsdl:output name="deleteDocumentResponse"
           message="impl:deleteDocumentResponse"/>
       </wsdl:operation>
       <wsdl:operation name="deleteDocument" parameterOrder="document">
          <wsdl:input name="deleteDocumentRequest1" message="impl:deleteDocumentRequest1"/>
          <wsdl:output name="deleteDocumentResponse1"
           message="impl:deleteDocumentResponse1"/>
       </wsdl:operation>
       <wsdl:operation name="getAllDocuments">
          <wsdl:input name="getAllDocumentsRequest" message="impl:getAllDocumentsRequest"/>
          <wsdl:output name="getAllDocumentsResponse"
           message="impl:getAllDocumentsResponse"/>
       </wsdl:operation>
       <wsdl:operation name="setDocumentState" parameterOrder="documentID stateID">
          <wsdl:input name="setDocumentStateRequest"
           message="impl:setDocumentStateRequest"/>
          <wsdl:output name="setDocumentStateResponse"
           message="impl:setDocumentStateResponse"/>
       </wsdl:operation>
       <wsdl:operation name="getAllState">
```

```
         <wsdl:input name="getAllStateRequest" message="impl:getAllStateRequest"/>
         <wsdl:output name="getAllStateResponse" message="impl:getAllStateResponse"/>
      </wsdl:operation>
      <wsdl:operation name="getDocumentUri" parameterOrder="documentID">
         <wsdl:input name="getDocumentUriRequest" message="impl:getDocumentUriRequest"/>
         <wsdl:output name="getDocumentUriResponse"
       message="impl:getDocumentUriResponse"/>
      </wsdl:operation>
      <wsdl:operation name="getDocumentInfo" parameterOrder="documentID">
         <wsdl:input name="getDocumentInfoRequest" message="impl:getDocumentInfoRequest"/>
         <wsdl:output name="getDocumentInfoResponse"
          message="impl:getDocumentInfoResponse"/>
      </wsdl:operation>
      <wsdl:operation name="getDocumentUriReadOnly" parameterOrder="documentID">
         <wsdl:input name="getDocumentUriReadOnlyRequest"
          message="impl:getDocumentUriReadOnlyRequest"/>
         <wsdl:output name="getDocumentUriReadOnlyResponse"
          message="impl:getDocumentUriReadOnlyResponse"/>
      </wsdl:operation>
   </wsdl:portType>
   <wsdl:binding name="SimpleDocumentManagerSoapBinding"
    type="impl:SimpleDocumentManager">
      <wsdlsoap:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http"/>
      <wsdl:operation name="loadDocument">
         <wsdlsoap:operation soapAction=""/>
         <wsdl:input name="loadDocumentRequest">
            <wsdlsoap:body use="encoded"
             encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
             namespace="http://documentManager.doc"/>
         </wsdl:input>
         <wsdl:output name="loadDocumentResponse">
            <wsdlsoap:body use="encoded"
             encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
             namespace="http://documentManager.doc"/>
         </wsdl:output>
      </wsdl:operation>
      <wsdl:operation name="addDocument">
         <wsdlsoap:operation soapAction=""/>
         <wsdl:input name="addDocumentRequest">
            <wsdlsoap:body use="encoded"
             encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
             namespace="http://documentManager.doc"/>
         </wsdl:input>
         <wsdl:output name="addDocumentResponse">
            <wsdlsoap:body use="encoded"
             encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
             namespace="http://documentManager.doc"/>
         </wsdl:output>
      </wsdl:operation>
      <wsdl:operation name="storeDocument">
         <wsdlsoap:operation soapAction=""/>
         <wsdl:input name="storeDocumentRequest">
            <wsdlsoap:body use="encoded"
             encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
             namespace="http://documentManager.doc"/>
         </wsdl:input>
         <wsdl:output name="storeDocumentResponse">
            <wsdlsoap:body use="encoded"
             encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
             namespace="http://documentManager.doc"/>
         </wsdl:output>
      </wsdl:operation>
      <wsdl:operation name="storeDocument">
         <wsdlsoap:operation soapAction=""/>
         <wsdl:input name="storeDocumentRequest1">
            <wsdlsoap:body use="encoded"
             encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
             namespace="http://documentManager.doc"/>
         </wsdl:input>
         <wsdl:output name="storeDocumentResponse1">
            <wsdlsoap:body use="encoded"
             encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
             namespace="http://documentManager.doc"/>
         </wsdl:output>
      </wsdl:operation>
      <wsdl:operation name="loadDocumentReadOnly">
         <wsdlsoap:operation soapAction=""/>
         <wsdl:input name="loadDocumentReadOnlyRequest">
```

```
            <wsdlsoap:body use="encoded"
             encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
             namespace="http://documentManager.doc"/>
         </wsdl:input>
         <wsdl:output name="loadDocumentReadOnlyResponse">
            <wsdlsoap:body use="encoded"
             encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
             namespace="http://documentManager.doc"/>
         </wsdl:output>
      </wsdl:operation>
      <wsdl:operation name="deleteDocument">
         <wsdlsoap:operation soapAction=""/>
         <wsdl:input name="deleteDocumentRequest">
            <wsdlsoap:body use="encoded"
             encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
             namespace="http://documentManager.doc"/>
         </wsdl:input>
         <wsdl:output name="deleteDocumentResponse">
            <wsdlsoap:body use="encoded"
             encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
             namespace="http://documentManager.doc"/>
         </wsdl:output>
      </wsdl:operation>
      <wsdl:operation name="deleteDocument">
         <wsdlsoap:operation soapAction=""/>
         <wsdl:input name="deleteDocumentRequest1">
            <wsdlsoap:body use="encoded"
             encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
             namespace="http://documentManager.doc"/>
         </wsdl:input>
         <wsdl:output name="deleteDocumentResponse1">
            <wsdlsoap:body use="encoded"
             encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
             namespace="http://documentManager.doc"/>
         </wsdl:output>
      </wsdl:operation>
      <wsdl:operation name="getAllDocuments">
         <wsdlsoap:operation soapAction=""/>
         <wsdl:input name="getAllDocumentsRequest">
            <wsdlsoap:body use="encoded"
             encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
             namespace="http://documentManager.doc"/>
         </wsdl:input>
         <wsdl:output name="getAllDocumentsResponse">
            <wsdlsoap:body use="encoded"
             encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
             namespace="http://documentManager.doc"/>
         </wsdl:output>
      </wsdl:operation>
      <wsdl:operation name="setDocumentState">
         <wsdlsoap:operation soapAction=""/>
         <wsdl:input name="setDocumentStateRequest">
            <wsdlsoap:body use="encoded"
             encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
             namespace="http://documentManager.doc"/>
         </wsdl:input>
         <wsdl:output name="setDocumentStateResponse">
            <wsdlsoap:body use="encoded"
             encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
             namespace="http://documentManager.doc"/>
         </wsdl:output>
      </wsdl:operation>
      <wsdl:operation name="getAllState">
         <wsdlsoap:operation soapAction=""/>
         <wsdl:input name="getAllStateRequest">
            <wsdlsoap:body use="encoded"
             encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
             namespace="http://documentManager.doc"/>
         </wsdl:input>
         <wsdl:output name="getAllStateResponse">
            <wsdlsoap:body use="encoded"
             encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
             namespace="http://documentManager.doc"/>
         </wsdl:output>
      </wsdl:operation>
      <wsdl:operation name="getDocumentUri">
         <wsdlsoap:operation soapAction=""/>
         <wsdl:input name="getDocumentUriRequest">
```

```
            <wsdlsoap:body use="encoded"
             encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
             namespace="http://documentManager.doc"/>
         </wsdl:input>
         <wsdl:output name="getDocumentUriResponse">
            <wsdlsoap:body use="encoded"
         encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
         namespace="http://documentManager.doc"/>
         </wsdl:output>
      </wsdl:operation>
      <wsdl:operation name="getDocumentInfo">
         <wsdlsoap:operation soapAction=""/>
         <wsdl:input name="getDocumentInfoRequest">
            <wsdlsoap:body use="encoded"
             encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
             namespace="http://documentManager.doc"/>
         </wsdl:input>
         <wsdl:output name="getDocumentInfoResponse">
            <wsdlsoap:body use="encoded"
             encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
             namespace="http://documentManager.doc"/>
         </wsdl:output>
      </wsdl:operation>
      <wsdl:operation name="getDocumentUriReadOnly">
         <wsdlsoap:operation soapAction=""/>
         <wsdl:input name="getDocumentUriReadOnlyRequest">
            <wsdlsoap:body use="encoded"
             encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
             namespace="http://documentManager.doc"/>
         </wsdl:input>
         <wsdl:output name="getDocumentUriReadOnlyResponse">
            <wsdlsoap:body use="encoded"
             encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
             namespace="http://documentManager.doc"/>
         </wsdl:output>
      </wsdl:operation>
   </wsdl:binding>
   <wsdl:service name="SimpleDocumentManagerService">
      <wsdl:port name="SimpleDocumentManager"
       binding="impl:SimpleDocumentManagerSoapBinding">
         <wsdlsoap:address
          location="http://localhost:8080/dms/services/SimpleDocumentManager"/>
      </wsdl:port>
   </wsdl:service>
</wsdl:definitions>
```

Listing A-1: WSDL der SimpleDocumentManager-Klasse

A.2 Schnittstellenbeschreibung der ComplexDocumentManager-Klasse

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions targetNamespace="http://documentManager.doc"
 xmlns:impl="http://documentManager.doc" xmlns:intf="http://documentManager.doc"
 xmlns:apachesoap="http://xml.apache.org/xml-soap"
 xmlns:wsdlsoap="http://schemas.xmlsoap.org/wsdl/soap/"
 xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:tns1="http://types.axis.apache.org"
 xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" xmlns="http://schemas.xmlsoap.org/wsdl/">
 <wsdl:types>
  <schema xmlns="http://www.w3.org/2001/XMLSchema"
   targetNamespace="http://types.axis.apache.org">
   <import namespace="http://schemas.xmlsoap.org/soap/encoding/"/>
   <complexType name="URI">
    <sequence>
     <element name="fragment" nillable="true" type="xsd:string"/>
     <element name="genericURI" type="xsd:boolean"/>
     <element name="host" nillable="true" type="xsd:string"/>
     <element name="path" nillable="true" type="xsd:string"/>
     <element name="port" type="xsd:int"/>
     <element name="queryString" nillable="true" type="xsd:string"/>
     <element name="scheme" nillable="true" type="xsd:string"/>
     <element name="schemeSpecificPart" nillable="true" type="xsd:string"/>
     <element name="userinfo" nillable="true" type="xsd:string"/>
    </sequence>
   </complexType>
  </schema>
 </wsdl:types>
   <wsdl:message name="checkOutURIRequest">
      <wsdl:part name="documentID" type="xsd:int"/>
   </wsdl:message>
   <wsdl:message name="checkInRequest1">
      <wsdl:part name="documentID" type="xsd:int"/>
      <wsdl:part name="uri" type="tns1:URI"/>
   </wsdl:message>
   <wsdl:message name="addDocumentToContainerResponse">
      <wsdl:part name="addDocumentToContainerReturn" type="xsd:boolean"/>
   </wsdl:message>
   <wsdl:message name="checkInRequest">
      <wsdl:part name="documentID" type="xsd:int"/>
      <wsdl:part name="file" type="xsd:anyType"/>
   </wsdl:message>
   <wsdl:message name="createContainerRequest">
      <wsdl:part name="name" type="xsd:string"/>
   </wsdl:message>
   <wsdl:message name="checkOutFileRequest">
      <wsdl:part name="documentID" type="xsd:int"/>
   </wsdl:message>
   <wsdl:message name="createContainerResponse">
      <wsdl:part name="createContainerReturn" type="xsd:anyType"/>
   </wsdl:message>
   <wsdl:message name="addDocumentToContainerRequest">
      <wsdl:part name="containerID" type="xsd:int"/>
      <wsdl:part name="documentID" type="xsd:int"/>
   </wsdl:message>
   <wsdl:message name="checkOutURIResponse">
      <wsdl:part name="checkOutURIReturn" type="tns1:URI"/>
   </wsdl:message>
   <wsdl:message name="removeDocumentFromContainerResponse">
      <wsdl:part name="removeDocumentFromContainerReturn" type="xsd:boolean"/>
   </wsdl:message>
   <wsdl:message name="removeDocumentFromContainerRequest">
      <wsdl:part name="containerID" type="xsd:int"/>
      <wsdl:part name="documentID" type="xsd:int"/>
   </wsdl:message>
   <wsdl:message name="checkInResponse">
      <wsdl:part name="checkInReturn" type="xsd:boolean"/>
   </wsdl:message>
   <wsdl:message name="checkOutFileResponse">
      <wsdl:part name="checkOutFileReturn" type="xsd:anyType"/>
   </wsdl:message>
   <wsdl:message name="checkInResponse1">
      <wsdl:part name="checkInReturn" type="xsd:boolean"/>
```

```
    </wsdl:message>
    <wsdl:portType name="ComplexDocumentManager">
       <wsdl:operation name="checkIn" parameterOrder="documentID file">
          <wsdl:input name="checkInRequest" message="impl:checkInRequest"/>
          <wsdl:output name="checkInResponse" message="impl:checkInResponse"/>
       </wsdl:operation>
       <wsdl:operation name="checkIn" parameterOrder="documentID uri">
          <wsdl:input name="checkInRequest1" message="impl:checkInRequest1"/>
          <wsdl:output name="checkInResponse1" message="impl:checkInResponse1"/>
       </wsdl:operation>
       <wsdl:operation name="createContainer" parameterOrder="name">
          <wsdl:input name="createContainerRequest" message="impl:createContainerRequest"/>
          <wsdl:output name="createContainerResponse"
           message="impl:createContainerResponse"/>
       </wsdl:operation>
       <wsdl:operation name="checkOutFile" parameterOrder="documentID">
          <wsdl:input name="checkOutFileRequest" message="impl:checkOutFileRequest"/>
          <wsdl:output name="checkOutFileResponse" message="impl:checkOutFileResponse"/>
       </wsdl:operation>
       <wsdl:operation name="checkOutURI" parameterOrder="documentID">
          <wsdl:input name="checkOutURIRequest" message="impl:checkOutURIRequest"/>
          <wsdl:output name="checkOutURIResponse" message="impl:checkOutURIResponse"/>
       </wsdl:operation>
       <wsdl:operation name="addDocumentToContainer" parameterOrder="containerID
        documentID">
          <wsdl:input name="addDocumentToContainerRequest"
           message="impl:addDocumentToContainerRequest"/>
          <wsdl:output name="addDocumentToContainerResponse"
           message="impl:addDocumentToContainerResponse"/>
       </wsdl:operation>
       <wsdl:operation name="removeDocumentFromContainer" parameterOrder="containerID
        documentID">
          <wsdl:input name="removeDocumentFromContainerRequest"
           message="impl:removeDocumentFromContainerRequest"/>
          <wsdl:output name="removeDocumentFromContainerResponse"
           message="impl:removeDocumentFromContainerResponse"/>
       </wsdl:operation>
    </wsdl:portType>
    <wsdl:binding name="ComplexDocumentManagerSoapBinding"
     type="impl:ComplexDocumentManager">
       <wsdlsoap:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http"/>
       <wsdl:operation name="checkIn">
          <wsdlsoap:operation soapAction=""/>
          <wsdl:input name="checkInRequest">
             <wsdlsoap:body use="encoded"
              encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
              namespace="http://documentManager.doc"/>
          </wsdl:input>
          <wsdl:output name="checkInResponse">
             <wsdlsoap:body use="encoded"
              encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
              namespace="http://documentManager.doc"/>
          </wsdl:output>
       </wsdl:operation>
       <wsdl:operation name="checkIn">
          <wsdlsoap:operation soapAction=""/>
          <wsdl:input name="checkInRequest1">
             <wsdlsoap:body use="encoded"
              encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
              namespace="http://documentManager.doc"/>
          </wsdl:input>
          <wsdl:output name="checkInResponse1">
             <wsdlsoap:body use="encoded"
              encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
              namespace="http://documentManager.doc"/>
          </wsdl:output>
       </wsdl:operation>
       <wsdl:operation name="createContainer">
          <wsdlsoap:operation soapAction=""/>
          <wsdl:input name="createContainerRequest">
             <wsdlsoap:body use="encoded"
              encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
              namespace="http://documentManager.doc"/>
          </wsdl:input>
          <wsdl:output name="createContainerResponse">
             <wsdlsoap:body use="encoded"
              encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
              namespace="http://documentManager.doc"/>
```

```
         </wsdl:output>
      </wsdl:operation>
      <wsdl:operation name="checkOutFile">
         <wsdlsoap:operation soapAction=""/>
         <wsdl:input name="checkOutFileRequest">
            <wsdlsoap:body use="encoded"
             encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
             namespace="http://documentManager.doc"/>
         </wsdl:input>
         <wsdl:output name="checkOutFileResponse">
            <wsdlsoap:body use="encoded"
             encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
             namespace="http://documentManager.doc"/>
         </wsdl:output>
      </wsdl:operation>
      <wsdl:operation name="checkOutURI">
         <wsdlsoap:operation soapAction=""/>
         <wsdl:input name="checkOutURIRequest">
            <wsdlsoap:body use="encoded"
             encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
             namespace="http://documentManager.doc"/>
         </wsdl:input>
         <wsdl:output name="checkOutURIResponse">
            <wsdlsoap:body use="encoded"
             encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
             namespace="http://documentManager.doc"/>
         </wsdl:output>
      </wsdl:operation>
      <wsdl:operation name="addDocumentToContainer">
         <wsdlsoap:operation soapAction=""/>
         <wsdl:input name="addDocumentToContainerRequest">
            <wsdlsoap:body use="encoded"
             encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
             namespace="http://documentManager.doc"/>
         </wsdl:input>
         <wsdl:output name="addDocumentToContainerResponse">
            <wsdlsoap:body use="encoded"
             encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
             namespace="http://documentManager.doc"/>
         </wsdl:output>
      </wsdl:operation>
      <wsdl:operation name="removeDocumentFromContainer">
         <wsdlsoap:operation soapAction=""/>
         <wsdl:input name="removeDocumentFromContainerRequest">
            <wsdlsoap:body use="encoded"
             encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
             namespace="http://documentManager.doc"/>
         </wsdl:input>
         <wsdl:output name="removeDocumentFromContainerResponse">
            <wsdlsoap:body use="encoded"
             encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
             namespace="http://documentManager.doc"/>
         </wsdl:output>
      </wsdl:operation>
   </wsdl:binding>
   <wsdl:service name="ComplexDocumentManagerService">
      <wsdl:port name="ComplexDocumentManager"
       binding="impl:ComplexDocumentManagerSoapBinding">
         <wsdlsoap:address
          location="http://localhost:8080/dms/services/ComplexDocumentManager"/>
      </wsdl:port>
   </wsdl:service>
</wsdl:definitions>
```

Listing A-2: WSDL der ComplexDocumentManager -Klasse

A.3 Web Services Graphical User Interface (WSGUI) Spezifikation

```
<?xml version="1.0" encoding="UTF-8" ?>
<schema xmlns="http://www.w3.org/2001/XMLSchema"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 xmlns:wsgui="http://fxagents.stanford.edu/2002/10/wsgui"
 targetNamespace="http://fxagents.stanford.edu/2002/10/wsgui"
 xmlns:xsl="http://www.w3.org/1999/XSL/Transform" elementFormDefault="qualified">
  <xsd:import namespace="http://www.w3.org/1999/XSL/Transform" schemaLocation="xslt.xsd"/>
  <element name="deployment">
    <annotation>
      <documentation>The top element of GUI Deployment Descriptor
       (*.guidd).</documentation>
    </annotation>
    <complexType>
      <all>
        <element ref="wsgui:wsdl" />
        <element ref="wsgui:stylesheets" />
        <element ref="wsgui:pages" />
        <element ref="wsgui:operations" />
        <element ref="wsgui:formComponents" />
        <element ref="wsgui:dynamicEnumerations" />
      </all>
    </complexType>
  </element>
  <element name="wsdl">
    <complexType>
      <attribute name="href" type="xsd:anyURI" use="required" />
    </complexType>
  </element>
  <element name="stylesheets">
    <complexType>
      <sequence>
        <element name="stylesheet" minOccurs="1" maxOccurs="unbounded">
          <complexType>
            <attribute name="name" type="xsd:Name" use="required" />
            <attribute name="href" type="xsd:anyURI" use="required" />
          </complexType>
        </element>
      </sequence>
    </complexType>
  </element>
  <element name="pages">
    <complexType>
      <sequence>
        <element name="page" minOccurs="1" maxOccurs="unbounded">
          <complexType>
            <sequence>
              <element name="prettyName" type="xsd:string" />
              <element name="description" type="xsd:string" />
            </sequence>
            <attribute name="name" type="xsd:Name" use="required" />
            <attribute name="stylesheet" type="xsd:Name" use="required" />
            <attribute name="operations" use="required">
              <simpleType>
                <list itemType="xsd:Name" />
              </simpleType>
            </attribute>
          </complexType>
        </element>
      </sequence>
    </complexType>
  </element>
  <element name="operations">
    <complexType>
      <sequence>
        <element name="operation" minOccurs="1" maxOccurs="unbounded">
          <complexType>
            <sequence>
              <element name="prettyName" type="xsd:string" />
              <element name="description" type="xsd:string" />
              <element ref="wsgui:submit" />
            </sequence>
            <attribute name="name" type="xsd:Name" use="required" />
```

```
            <attributeGroup ref="wsgui:portTypeOperation" />
          </complexType>
        </element>
      </sequence>
    </complexType>
  </element>
  <attributeGroup name="portTypeOperation">
    <attribute name="wsdlService" type="xsd:QName" use="required" />
    <attribute name="wsdlPort" type="xsd:NCName" use="required" />
    <attribute name="wsdlOperation" type="xsd:NCName" use="required" />
  </attributeGroup>
  <element name="formComponents">
    <complexType>
      <sequence>
        <element name="formComponent" type="wsgui:formComponent_type" minOccurs="1"
         maxOccurs="unbounded" />
      </sequence>
    </complexType>
  </element>
  <complexType name="formComponent_type">
    <choice>
      <element name="input" type="wsgui:input_type" />
      <element name="secret" type="wsgui:input_type" />
      <element name="select1" type="wsgui:select1_type" />
    </choice>
    <attribute name="xpath" type="xsd:string" />
    <attribute name="name" type="xsd:Name" />
    <attribute name="subquery" type="xsd:string" />
  </complexType>
  <complexType name="input_type">
    <sequence>
      <element ref="wsgui:label" />
      <element ref="wsgui:help" minOccurs="0" />
    </sequence>
    <attribute name="class" type="xsd:string" />
  </complexType>
  <complexType name="select1_type">
    <sequence>
      <element ref="wsgui:label" />
      <element ref="wsgui:item" minOccurs="0" maxOccurs="unbounded" />
      <element ref="wsgui:help" minOccurs="0" />
    </sequence>
    <attribute name="class" type="xsd:string" />
    <attribute name="dynamicEnumeration" type="xsd:Name" />
    <attribute name="appearance">
      <simpleType>
        <restriction base="xsd:string">
          <enumeration value="minimal" />
          <enumeration value="compact" />
        </restriction>
      </simpleType>
    </attribute>
    <attribute name="navigation">
      <simpleType>
        <restriction base="xsd:string">
          <enumeration value="link" />
        </restriction>
      </simpleType>
    </attribute>
  </complexType>
  <element name="submit">
    <complexType>
      <sequence>
        <element ref="wsgui:label" />
      </sequence>
      <attribute name="class" type="xsd:string" />
    </complexType>
  </element>
  <element name="label">
    <simpleType>
      <restriction base="xsd:string" />
    </simpleType>
  </element>
  <element name="help">
    <simpleType>
      <restriction base="xsd:string" />
    </simpleType>
  </element>
```

```
  <element name="item">
    <complexType>
      <sequence>
        <element name="label" type="xsd:string" />
        <element name="value" type="xsd:string" />
      </sequence>
    </complexType>
  </element>
  <element name="dynamicEnumerations">
    <complexType>
      <sequence>
        <element name="dynamicEnumeration" type="wsgui:dynamicEnumeration_type"
         minOccurs="1" maxOccurs="unbounded" />
      </sequence>
    </complexType>
  </element>
  <complexType name="dynamicEnumeration_type">
    <sequence>
      <element ref="wsgui:baseOperation" />
      <element ref="wsgui:parameters" minOccurs="0" />
      <element name="inputPartTransformer" type="wsgui:partTransformer" minOccurs="0"
       maxOccurs="unbounded" />
      <element name="outputPartTransformer" type="wsgui:partTransformer" minOccurs="0"
       maxOccurs="unbounded" />
    </sequence>
    <attribute name="name" type="xsd:Name" use="required" />
    <attribute name="range" use="required">
      <simpleType>
        <restriction base="xsd:string">
          <enumeration value="all" />
          <enumeration value="sub" />
        </restriction>
      </simpleType>
    </attribute>
  </complexType>
  <element name="baseOperation">
    <complexType>
      <attributeGroup ref="wsgui:portTypeOperation" />
    </complexType>
  </element>
  <element name="parameters">
    <complexType>
      <sequence>
        <element name="parameter" minOccurs="1" maxOccurs="unbounded">
          <complexType>
            <attribute name="name" type="xsd:string" use="required" />
            <attribute name="formComponent" type="xsd:string" use="required" />
          </complexType>
        </element>
       </sequence>
    </complexType>
  </element>
  <complexType name="partTransformer">
    <sequence>
      <element ref="xsl:stylesheet" />
    </sequence>
    <attribute name="part" type="xsd:NCName" use="required" />
  </complexType>
</schema>
```

Listing A-3: WSGUI.xsd [Spi05]

A.4 Implementierung des WSGUIPortlet

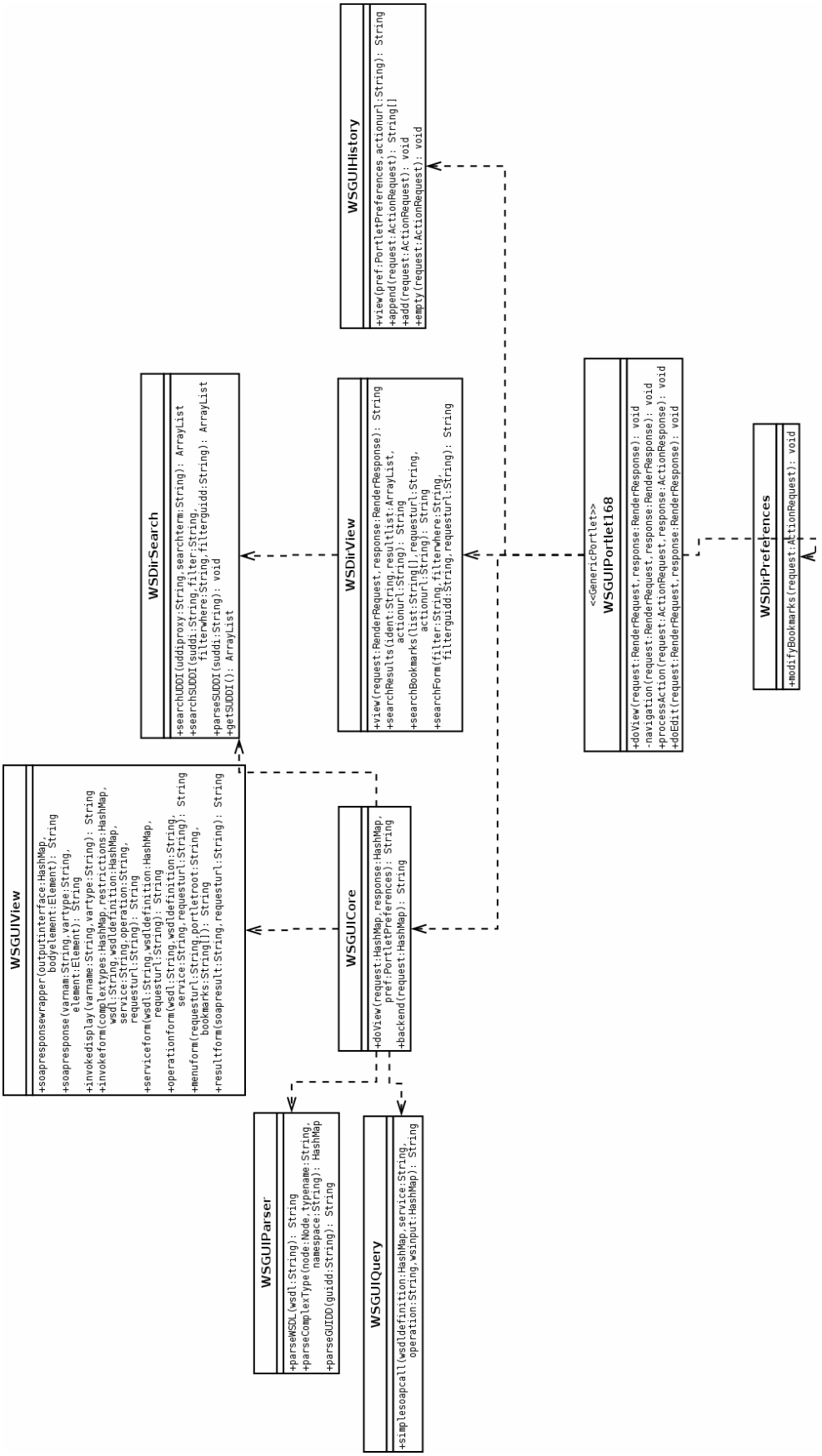

Abbildung A-1: UML-Klassendiagramm von WSGUIPortlet

SELBSTÄNDIGKEITSERKLÄRUNG

Hiermit versichere ich, die vorliegende Arbeit selbständig, ohne fremde Hilfe und ohne Benutzung anderer als der von mir angegebenen Quellen angefertigt zu haben. Alle aus fremden Quellen direkt oder indirekt übernommenen Gedanken sind als solche gekennzeichnet. Die Arbeit wurde noch keiner Prüfungsbehörde in gleicher oder ähnlicher Form vorgelegt.

Dresden, den 10.08.2005